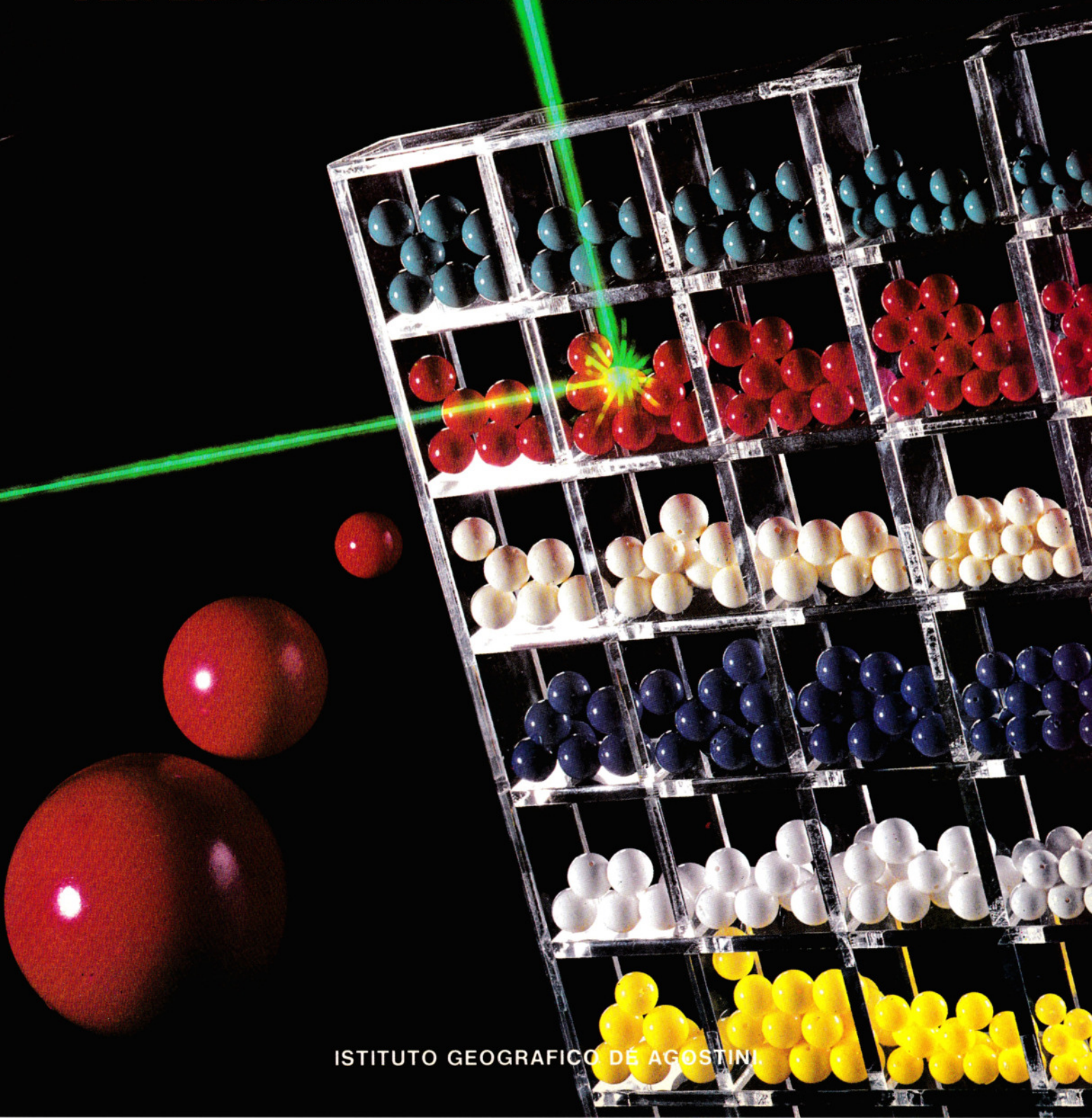
INPUT
9
L. 2800
CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER
ISTITUTO GEOGRAFICO DE AGOSTINI

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

Pubblicazione a fascicoli settimanali
edita dall'Istituto Geografico De Agostini

volume I - fascicolo 9

Direttori: Achille Boroli - Adolfo Boroli

Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

---

Redazione dell'edizione italiana a cura della:
Logical Studio Communication

Traduzione dall'inglese a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo

Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini

Direzione:
Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5

Redazione:
Milano (20149), via Mosè Bianchi 6 - tel. (02) 4694451

Amministrazione, abbonamenti e servizio arretrati:
Istituto Geografico De Agostini - Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5.

Copertine e risguardi per i volumi dell'opera saranno messi in vendita a L. 6000 la copia (L. 7500 all'estero).

Le copie arretrate saranno disponibili per un anno dal completamento dell'opera e potranno essere prenotate nelle edicole o direttamente presso l'Editore. Per i fascicoli arretrati, trascorse 12 settimane dalla loro pubblicazione, è applicato un sovrapprezzo di L. 400 sul prezzo di copertina in vigore al momento dell'evasione dell'ordine. Spedizione contro rimessa di pagamento anticipato; non vengono effettuate spedizioni contrassegno.

L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

Registrato presso il Tribunale di Novara n. 11 in data 19-5-1984.

Direttore responsabile: Emilio Bucciotti

Spedizione in abbonamento postale Gruppo II/70 (Autorizzazione della Direzione provinciale delle PP.TT. di Novara).

Distribuzione A. & G. Marco - Milano, via Fortezza 27 - tel. (02) 2526.

Pubblicazione a fascicoli settimanali. Esce il martedì.

Stampato in Italia - I.G.D.A. Officine Grafiche, Novara - 188412.

Referenze dei disegni e delle fotografie:
Copertina: Dave King. Pag. 257 Nick Farmer. Pag. 258 K. Goebel/ZEFA/Hussein Hussein. Pagg. 260, 261 Ray Duns. Pag. 262 Tony Stone/Hussein Hussein. Pagg. 264, 266, 267 Alan Baker. Pagg. 269, 270, 272 Dave King. Pagg. 277, 279 Nick Farmer/Nick Mijnheer. Pag. 283 Digital Arts. Pagg. 284, 286, 288 Peter Bentley.

APPLICAZIONI 5

VISUALIZZARE FATTI E CIFRE **257**

Questo programma produce grafici per un immediato esame di un gruppo di dati

GIOCHI AL COMPUTER 9

LO SVILUPPO DI UN'AVVENTURA **264**

La prima parte di una serie dedicata al completo sviluppo di un gioco del tipo "adventure"

PROGRAMMAZIONE BASIC 19

TIRO INCROCIATO CON LE MATRICI **269**

Le matrici a più dimensioni, offerte dal BASIC, consentono di memorizzare efficacemente i dati

CODICE MACCHINA 10

COME IMMETTERE IL CODICE MACCHINA **276**

Come risparmiare fatica nel redigere piccoli programmi

PROGRAMMAZIONE BASIC 20

RELAZIONI... SENSATE! **284**

Un corretto uso degli operatori logici e aritmetici permette di valutare le espressioni matematiche

Con questo fascicolo termina il primo volume di

INPUT

Lo potete rilegare con l'apposita copertina posta in vendita in corrispondenza con l'uscita del fascicolo n. 7. Per raccogliere e poi rilegare i fascicoli del secondo volume prenotate subito una nuova copertina:

LE SPECIALI COPERTINE INTERCAMBIABILI SONO SEMPRE DISPONIBILI

Chiedetele in qualsiasi edicola, oppure direttamente all'Editore, versando l'importo sul c/c postale n. 111286 intestato all'Istituto Geografico De Agostini di Novara e specificando la causale.

I 'trasferibili' e le relative istruzioni per contrassegnare i dorsi vengono forniti in ciascuna copertina.

Gli abbonati riceveranno le copertine senza farne richiesta essendo il relativo prezzo compreso nel canone di abbonamento

VISUALIZZARE FATTI E CIFRE

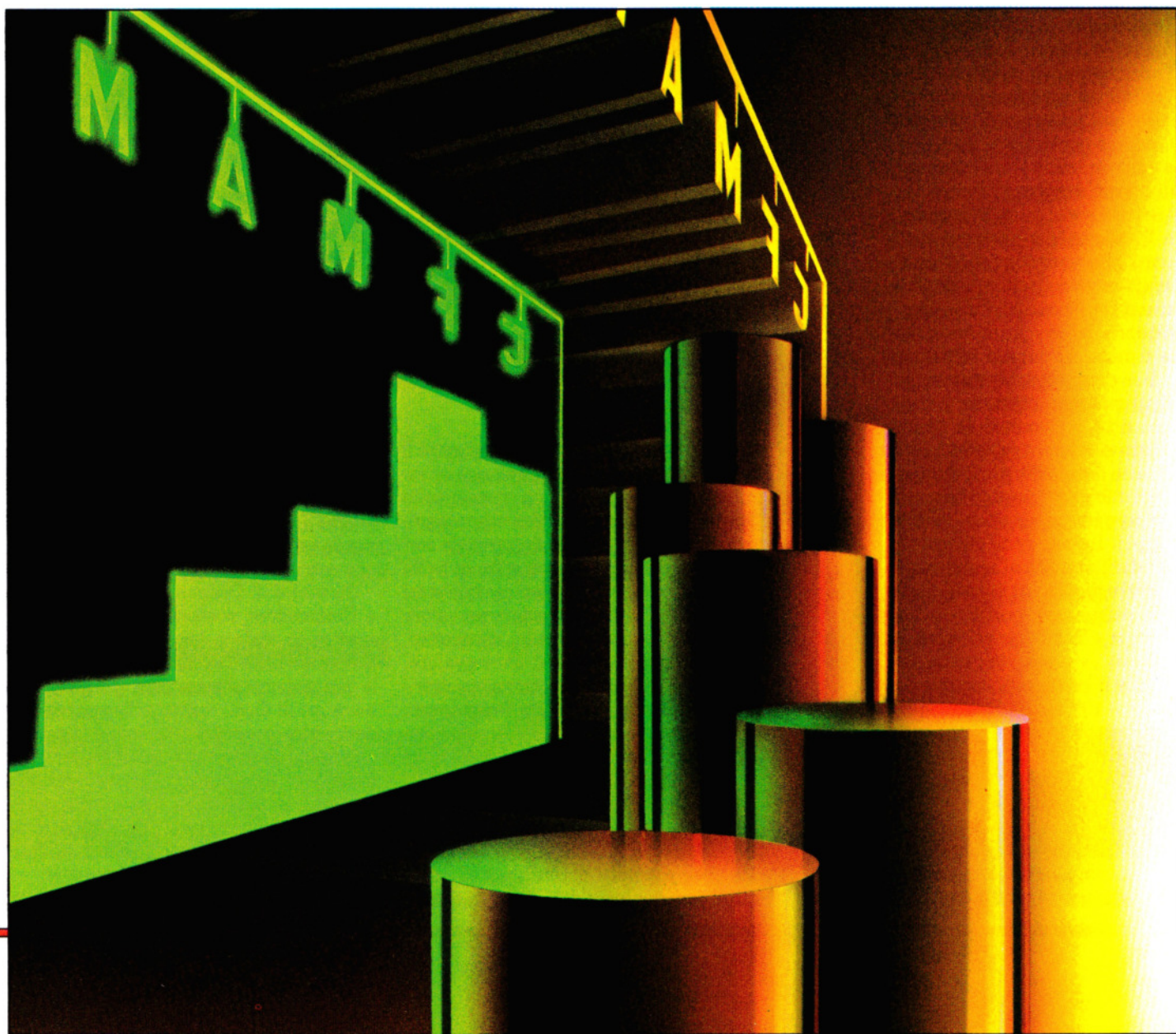
- A CHE SERVE IL PROGRAMMA
- IMMISSIONE DELLE INFORMAZIONI
- CORREZIONI DEI DATI
- DISEGNO DEL GRAFICO
- RIDUZIONE IN SCALA DEGLI ASSI

Per un'analisi istantanea, basta immettere i dati nel computer e osservare la loro trasformazione in un colorato istogramma dall'aspetto professionale

Nella pubblicità di computer per applicazioni commerciali, sono immancabilmente presenti complesse rappresentazioni di dati, tabulati sulle vendite, sulle scorte, e via dicendo. Questo genere di informazioni trova la sua migliore presentazione in forma di grafici, molto più facili da comprender a colpo d'occhio. Tracciare grafici a mano è un lavoro lungo e tedioso, mentre un computer professionale eccelle in questo campo, producendo istantaneamente immagini colorate. Tuttavia, questo non è un campo limitato solo ai grandi apparecchi, ma adatto anche per piccole applicazioni pratiche con gli home computer. Tutti quelli qui previsti, tranne il Vic 2Ø e lo ZX81, hanno possibilità grafiche e matematiche capaci di affrontare questo compito.

Il possessore di uno home computer ha raramente a che fare con le grandi masse di informazioni che anche piccole attività commerciali riescono a produrre, ma non per questo mancano occasioni per sfruttare utilmente il computer. Per esempio, un grafico relativo al bilancio domestico può risultare utile per valutare l'andamento di tutto un anno: si è speso di più, per il software del compuer (o per i giornali, gli svaghi e la macchina), in autunno o in estate? In quale periodo il risparmio ha superato un certo livello e per quanto tempo?

Oltre a queste applicazioni di tipo economico, ci sono svariati argomenti di in-

teresse generale, oppure temi e cifre relativi a un hobby, da analizzare e rappresentare.

Questi argomenti vanno, per esempio, dal numero di presenze in un locale pubblico all'analisi climatica dell'anno. Una raffigurazione grafica è altrettanto utile per i risultati sportivi o per valutare l'ampliarsi di una collezione.

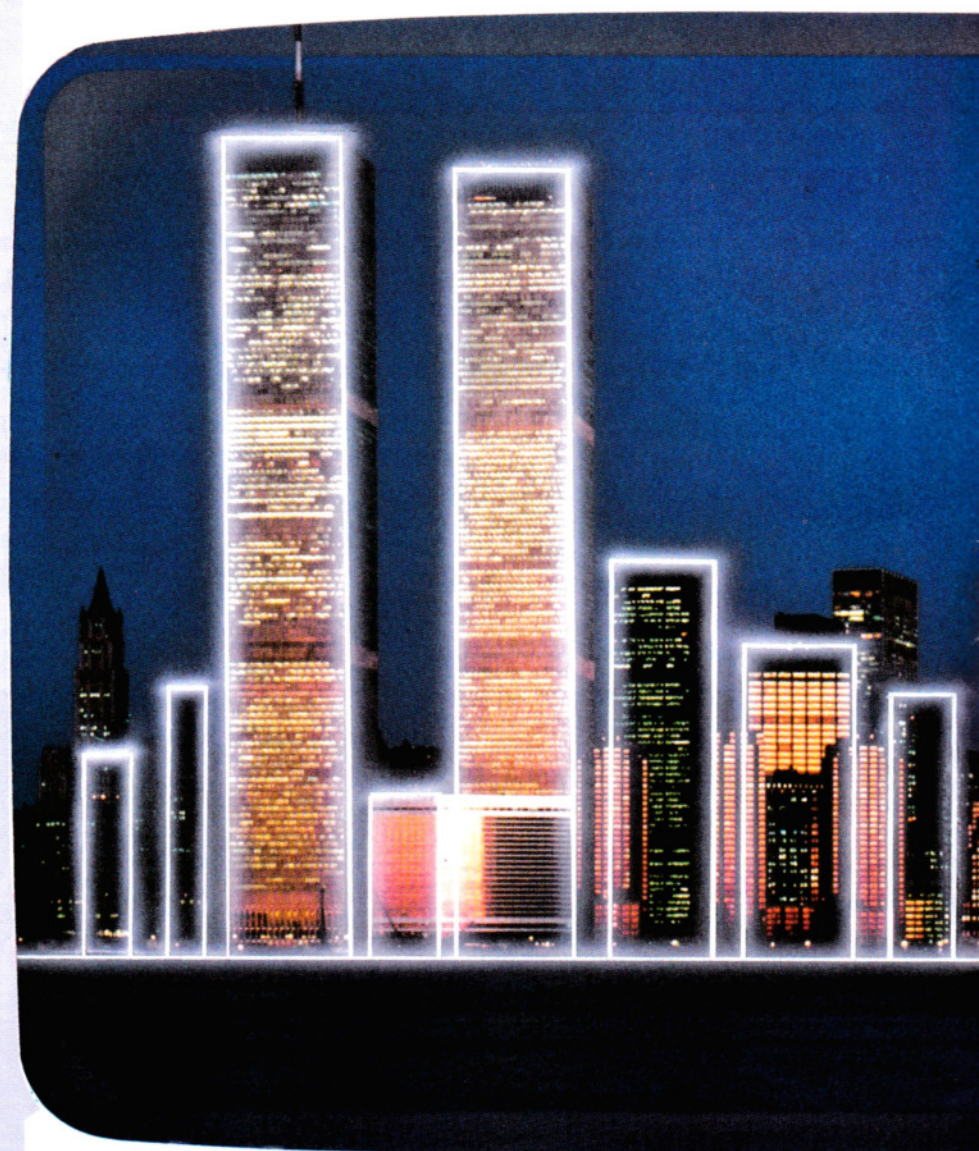
Il semplice programma qui presentato permette di preparare rapidamente la rappresentazione visiva di qualsiasi statistica, rispètto a un intervallo di tempo. L'arco massimo di valori che il programma può coprire dipende dal computer usato. Dragon e Tandy sono limitati entro +99 e −99; lo Spectrum, più o meno, fino a un migliaio, mentre i computer Acorn e Commodore manipolano numeri di ogni grandezza: unità, decine centinaia, migliaia o anche milioni, se i nostri conti lo richiedono.

USO DEL PROGRAMMA

Impartendo un RUN, viene visualizzato un 'menu' o lista di opzioni. Nel selezionare l'opzione per immettere nuovi dati, si è invitati ad associare un nome a ciascun asse, che poi apparirà sul grafico. Nell'introdurre questi nomi, si faccia attenzione a non confondersi tra i due assi: il numero di barre, destinate ai mesi, agli anni o a quel che sono, viene disegnato sull'asse x e il valore delle barre, siano lire o altre unità, viene riportato sull'asse y.

Viene poi chiesto il numero di barre da tracciare. Il numero massimo, di nuovo, dipende dal computer usato, o meglio dalle dimensioni del suo schermo grafico.

Per esempio, lo Spectrum può disegnare al massimo 25 barre, Dragon e Tandy fino a 26, il Commodore 64 fino a 3Ø. Gli Acorn sarebbero in effetti capaci di disegnarne quasi 2ØØ, ma il limite pratico è di circa 8Ø: oltre questo, le barre sottili sarebbero irregolari o troppo fini per essere visibili. Infine, viene chiesta l'immissione dei dati: accanto al numero di ogni barra occorre digitare il valore corrispondente, che può essere positivo o negativo. Su Dragon e Tandy questi valori possono andare da −999 a 999; sullo Spectrum da −1ØØØ a 1ØØØ; sugli Acorn e il Commodore da -10^{38} a 10^{38}. Se con lo Spectrum, il Dragon e il Tandy i dati escono dalla gamma consentita dal programma, l'ovvio rimedio è riportare i valori in unità di centinaia, migliaia, milioni o superiori se necessario.

Terminata l'immissione dell'ultimo valore, ricompare il menu e si può scegliere se correggere i dati (per cambiare valori immessi in modo incorretto) o visionare il grafico. Se si sceglie la correzione, o i valori saranno visualizzati a turno sullo schermo (Commodore e Acorn), oppure verrà chiesto il numero della barra da correggere (Spectrum, Dragon e Tandy). Chi usa il Commodore deve premere qualsiasi tasto meno RETURN per lasciare un valore inalterato e passare a quello successivo, oppure RETURN seguito dal nuovo valore. Chi ha un Acorn prema la barra spaziatrice e chi ha uno Spectrum, un Dragon o un Tandy segua le istruzioni che appaiono sullo schermo. Quando si è soddisfatti dei valori, si selezioni l'opzione per visionare il grafico. Con gli Acorn, si ha una sola possibilità di grafico, disegnato quasi all'istante. Sul BBC compare il fattore di scala (per esempio ×1ØØØ) per il quale moltiplicare i valori sull'asse y. Poiché tale scritta compare sulla prima linea sopra l'asse y, su alcune TV può non apparire. Per ovviare all'inconveniente, si digiti *TV255 RETURN prima di dare il RUN, ottenendo un abbassamento dell'immagine video corrispondente a una linea.

Gli utenti di Dragon, Tandy, Spectrum e Commodore possono scegliere tra un grafico in scala e uno a schermo pieno. L'opzione in scala visualizza il grafico con le dimensioni sull'asse y arrotondate a un massimo di dieci, cento, mille ecc., secondo il valore massimo dei dati e il grafico può non riempire tutto lo schermo. L'opzione a tutto schermo disegna il grafico sull'intera area dello schermo, ma visualizza gli effettivi valori dei dati (non in scala) sull'asse y. L'Acorn visualizza sempre un grafico a schermo pieno.

Per chiarezza, le barre disegnate dai

micro dell'Acorn, del Commodore e dello Spectrum sono separate da spazi o barre colorate. Non è possibile far ciò sul Dragon né sul Tandy, perciò le barre sono colorate alterantivamente in azzurro e giallo per valori positivi e in rosso e arancio per valori negativi.

```
10 LET g=0: POKE 23609,20: POKE
   23658,8
100 BORDER 7: PAPER 7: INK 0: CLS
110 PRINT BRIGHT 1; PAPER 3; INK 7; AT
   4,9;"□O□P□Z□I□O□N□I□"
120 PRINT BRIGHT 1;AT 7,6;"□1:-□
   IMMISSIONE□DATI□□□"
130 PRINT BRIGHT 1;AT 9,6;"□2:-□
   VISIONE/MODIFICA DATI□"
140 PRINT BRIGHT 1 ; AT 11,6; "□ 3 : -
   □ GRAFICO □ □ □ □ □ □"
145 PRINT BRIGHT 1;AT 13,6;"□4:-□
   GRAFICO SU SCHERMO INTERO"
150 PRINT BRIGHT 1; FLASH 1; INK 2; AT
   16,9;"□QUALE OPZIONE□"
160 IF INKEY$="" THEN GOTO 160
170 LET a$=INKEY$: IF a$<"1" OR a$
   >"4" THEN GOTO 160
175 IF a$<>"1" AND g=0 THEN GOTO
   160
180 GOSUB VAL a$*1000: GOTO 100
500 REM ** ROUTINE DI INPUT NUMERICO**
510 INPUT (w$); LINE a$: IF LEN a$=0
   THEN GOTO 510
520 FOR j=1 TO LEN a$
540 IF (a$(j)>="0" AND a$(j)<=
   "9") OR a$(j)="." OR a$(j)=
   "-" THEN NEXT j: LET v=VAL a$
   : RETURN
550 GOTO 510
1000 REM **ROUTINE DI INPUT**
1010 BORDER 1: PAPER 1: INK 7: CLS
1020 INPUT "NOME PER L'ASSE X?□"; LINE
   x$
1030 PRINT INVERSE 1;AT 0,0;"□"; x$;"□"
1040 INPUT "NOME PER L'ASSE Y?□"; LINE
   y$
1050 PRINT INVERSE 1;AT 0,16;"□"; y$;"□"
1060 LET w$="QUANTI□"+x$+"□
   (1-25)?□": GOSUB 500
1070 IF v<1 OR v>25 OR v<>INT v
   THEN GOTO 1060
1090 LET z=v:DIM a(z)
1100 FOR k=1 TO z
1110 LETw$="IMMETTERE DATI PER□"+
   STR$ k+"□": GOSUB 500
1120 LET a(k)=v
1130 PRINT k,a(k)
1140 NEXT k: LET g=1: PAUSE 50:
   RETURN
2000 REM **ROUTINE VISIONE/MODIFICA**
2010 BORDER 2: PAPER 2: INK 7
2020 LET cn=1
2025 CLS : PRINT PAPER 6; INK 2; AT
   0,0;x$,y$;TAB 31;"□"
2030 PRINT cn,a(cn)
2035 PRINT #1; PAPER 6; INK 2;AT
   0,0;"□□EDIT per modificare il
   valore□□□□□□o qualsiasi altro tasto
   per continuare□□□□"
2040 PAUSE 0
2050 IF INKEY$="" THEN GOTO 2050
2060 LET c$=INKEY$
2070 IF c$=CHR$ 7 THEN GOSUB 2500
2080 IF cn=z THEN PRINT PAPER 6; INK
   2;"FINE DEI DATI": PAUSE 100: RETURN
2090 LET cn=cn+1: IF cn=21 THEN
   GOTO 2025
2100 GOTO 2030
2500 LET w$="IMMETTERE NUOVO
   VALORE DI□"+STR$ cn+"□": GOSUB
   500
2510 LET a(cn)=v: PRINT PAPER 6; INK
   2;cn,a(cn);TAB 31;"□": RETURN
3000 REM **GRAFICO CON SCALA**
3010 BORDER 0: PAPER 0: INK 7: CLS: LET
   hi=0: LET lo=0
3020 FOR k=1 TO z
3030 IF a(k)>hi THEN LET hi=a(k)
3040 IF a(k)<lo THEN LET lo=a(k)
3050 NEXT k
3060 LET type=2: LET org=4
3070 IF lo<0 THEN LET type=1: LET org
   =84
3080 LET h=hi: IF ABS lo>hi THEN LET h
   =ABS lo
3090 LET ra=hi-lo
3100 IF h<=1 THEN LET hi=1: GOTO
   3150
3110 IF h<=10 THEN LET hi=10: GOTO
   3150
3120 IF h<=100 THEN LET hi=100:
```

Acorn: grafico delle oscillazioni stagionali di temperatura

Spectrum: profitti e perdite su 12 mesi

```
   GOTO 3150
3130 IF h<=1000 THEN LET hi=1000:
   GOTO 3150
3140 IF h<=10000 THEN LET hi=10000
3150 LET wd=INT (25/z)
3160 IF type=1 THEN LET ft=80/hi
3170 IF type=2 THEN LET ft=162/hi
3180 PLOT 56,org: DRAW 198,0
3190 PLOT 55,4: DRAW 0,160
3200 FOR n=4 TO 168 STEP 8: PLOT 52,n:
   DRAW 3,0: NEXT n
3220 PRINT #1; PAPER 1;AT 0,14;"□";x$;
   "□"
3225 LET z$="□"+y$+"□"
3230 FOR n=1 TO LEN z$
3240 PRINT AT n+(19-LEN y$)/2,0;
   PAPER 1;z$(n)
3250 NEXT n
3255 LET dc=1
3260 IF type=1 THEN GOTO 3320
3270 FOR n=hi TO 0 STEP -(hi/10)
3275 IF n<.01 THEN LET n=0
3280 LET n$=STR$ n
3290 PRINT AT dc,(6-LEN n$);n
3295 LET dc=dc+2
3300 NEXT n
3310 GOTO 3400
3320 FOR n=hi TO -hi STEP -(hi/5)
3330 IF n<.01 and n>-.01 THEN LET n
   =0
3340 LET n$=STR$ n
3350 PRINT AT dc, (6-LEN n$);n
3360 LET dc=dc+2
3370 NEXT n
3400 LET ink=1
3410 FOR n=1 TO z
3420 LET cm=org
3430 LET ink=ink+1: IF ink=8 THEN LET
   ink=2
3440 INK ink
3450 FOR m=1 TO (a(n)*ft) STEP SGN a(n)
3460 PLOT 56+(n-1)*wd*8,cm: DRAW
   wd*8-2,0
3470 LET cm=cm+SGN a(n)
3480 NEXT m
3490 NEXT n
3500 IF INKEY$<>"" THEN GOTO 3500
3510 IF INKEY$="" THEN GOTO 3510
3520 RETURN
4000 REM **GRAFICO SU SCHERMO
   INTERO**
4010 BORDER 0: PAPER 0: INK 7: CLS : LET
   hi=0: LET lo=0
4020 FOR n=1 TO z
4030 IF a(n)>hi THEN LET hi=a(n)
4040 IF a(n)<lo THEN LET lo=a(n)
4050 NEXT n
4060 LET ra=hi-lo: LET ft=175/ra: LET
   org=(ra-hi)*ft
4070 LET wd=INT (25/z)
4080 PLOT 56,org: DRAW 198,0
4090 PLOT 55,0: DRAW 0,175
4100 PRINT #1; PAPER 1;AT 0,14;"□";x$;
   "□"
4110 LET z$ = "□"+y$+"□"
4120 FOR n=1 TO LEN z$
4130 PRINT AT n+(19-LEN y$)/2,0;
   PAPER 1;z$(n)
4140 NEXT n
4150 PRINT AT 0,0;hi;AT 21,0;lo
4200 GOTO 3400
```

C=

```
0 BB=53280:GOTO 200
1 CLR:BB=53280:FF=1
2 POKE BB,0:POKE BB+1,0
3 Q$ = "■■■■■■■■■■■■
   ■■■■■■■■■■■■■■■■
   ■"
4 INPUT "♥NOME□DEL□GRAFICO";N$
5 INPUT "♥πNUMERO□DI□BARRE (1-
   30)";XX
6 IF XX<1 OR XX>30 THEN 5
7 DIM A(XX),B(XX),C(XX)
8 FOR Z=1 TO 30
9 IF XX*Z>30 THEN X1=Z-1: GOTO 10
10 NEXT Z
11 PRINT "♥IMMETTERE□DATI:■"
12 FOR Z=1 TO XX:PRINT "BARRA"; Z;:
   INPUT C(Z):NEXT Z
13 A=0:B=0:C=0:D=0
14 FOR Z=1 TO XX
15 IF C(Z)>A THEN A=C(Z)
16 IF C(Z)<B THEN B=C(Z)
17 IF C(Z)<0 THEN B(Z)=C(Z): A(Z)=0
18 IF C(Z)>0 THEN A(Z)=C(Z): B(Z)=0
19 NEXT Z
20 C=A+ABS(B)
22 GOTO 200
24 POKE BB,5:POKE BB+1,0
25 PRINT "♥"N$
26 FOR Z=2 TO 22
27 POKE 1033+Z*40,64:POKE 55305+
   Z*40,6
28 POKE 1032+Z*40,45:POKE 55304+
   Z*40,3
29 NEXT Z
30 FOR Z=1 TO XX:W=0:CO=8
31 IF B(Z)<0 THEN CO=2
32 FOR ZZ=B/D TO A/D
33 W=W+1:IF INT(ZZ)<>0 THEN 38
34 FOR G=1 TO X1
35 POKE (1953-X1)+(Z*X1+G)-W*40,64
```

Dragon e Tandy: un grafico di percentuali

Commodore 64: ogni barra simula un effetto 3D

```
36 POKE (56225 – X1) + (Z*X1 + G) – W*40,7
37 NEXT G:GOTO 43
38 FOR G = 1 TO X1:CA = 227:CC = CO: IF G
   < > 1 AND G = X1 THEN CO = 1
39 IF ZZ < 0 THEN CA = 228
40 IF B(Z)/D > ZZ OR A(Z)/D < ZZ THEN 43
41 POKE((1953 – X1) + Z*X1 + G) – W*40,CA:
   POKE((56225 – X1) + Z*X1 + G) – W*40,
   CO
42 CO = CC:NEXT G
43 POKE BB, RND(1)*2 + 5
44 NEXT ZZ,Z:POKE BB,5
45 PRINT "■ ■ ■"; LEFT$ (Q$, 20 –
   ABS (B/D))
46 FOR Z = 0 TO A STEP D*2: IF C = <
   10 AND INT(Z) < > Z THEN PRINT"■
   ■■": GOTO48
47 PRINT INT(Z)"■■■"
48 NEXT Z
49 PRINT "■■";LEFT$(Q$,20 – ABS(B/D))
50 FOR Z = 0 TO B STEP – D*2: IF C =
   < 10 AND INT(Z) < > Z THEN PRINT
   "■":GOTO 52
51 PRINT INT(Z)"■"
52 NEXT Z
53 POKE 198,0
54 GET A$:IF A$ = "" THEN 54
55 GOTO 200
100 POKE BB,1:POKE BB + 1,1
103 IF FF = 0 THEN 500
105 PRINT "■■ ■"TAB(10)"PLOTTER□DI
□GRAFICI□A□BARRE□"
110 PRINT "■■□□□■ ↑"
115 FOR Z = 1 TO 15
120 PRINT "□□□■":NEXT Z
125 PRINT "□□□■■■";
130 FOR Z = 1 TO 30
```

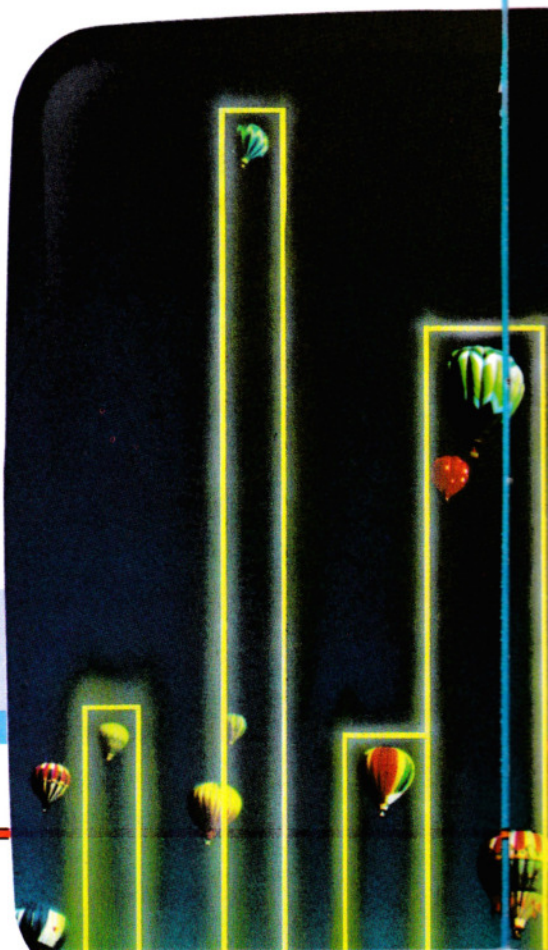
```
135 PRINT " – ";:NEXT Z
140 PRINT " > "
145 PRINT "■■■■■■■■■■■
■■■■VALORE:□UNITÀ"
150 PRINT "■■■■■■■■■■
■■■■■■BARRE:□SETT/MESI/
ANNI"
155 PRINT "■■■■■"TAB(9)
   "PREMERE□UN□TASTO□PER□
   CONTINUARE"
160 POKE 198,0
165 WAIT 198,1
170 GOTO 24
200 POKE BB,6:POKE BB + 1,6
210 PRINT "■■■■■■■■"TAB
   (15)"■■□□OPZIONI□□■■"
220 PRINT TAB(12)"1□IMMISSIONE□DATI"
230 PRINT TAB(12)"2□GRAFICO□CON□
   SCALA"
240 PRINT TAB(12)"3□VISIONE/MODIFICA□
   DATI"
245 PRINT TAB(12)"4□GRAFICO□SU□
   SCHERMO□INTERO"
250 PRINT TAB(13)"■■■QUALE□
   OPZIONE□?"
260 POKE 198,0
270 GET A$
280 ON VAL(A$) GOTO400,700,300, 600
290 GOTO 270
300 IF FF = 0 THEN 500
305 POKE BB,7:POKE BB + 1,7
310 PRINT "■■DATI□PER□"N$
315 PRINT,"■■ BARRE","■ ALTEZZA■"
320 FOR Z = 1 TO XX
325 IF A(Z) < > 0 THEN PRINT,"■"Z,
   "■"A(Z):GOTO 335
330 PRINT,"■"Z,"■"B(Z)
```

```
335 POKE 198,0
340 GET A$
345 IF A$ = "" THEN 340
350 IF A$ < > CHR$(13) THEN 370
355 PRINT "■■"TAB(18);
360 INPUT C(Z)
365 GOTO 13
370 NEXT Z
375 GOTO 200
400 POKE BB,13:POKE BB + 1,13
410 PRINT"■■■■■■■■■■
   ■■■■PREMERE□(■I
   ■)MMISSIONE DATI□,□(■M■)ENU"
420 GET A$
430 IF A$ = "I" THEN 1
440 IF A$ = "M" THEN 200
450 GOTO 420
500 PRINT"■■"TAB(16)"NESSUN□DATO!"
510 FOR Z = 1 TO 500:NEXT Z
520 GOTO 200
600 IF FF = 0 THEN 500
605 IF C < = 10 THEN D = .5:GOTO 100
606 IF C < = 20 THEN D = 1:GOTO 100
610 D = C/20:GOTO 100
700 IF FF = 0 THEN 500
705 IF C < = 10 THEN D = 1.25:GOTO 100
706 IF C < = 20 THEN D = 2.5:GOTO 100
710 D = C/20 + 9:D = D*.1:D = INT(D)*10:
   GOTO 100
```

```
10 A = Q
20 MODE1
30 VDU19,0,4,0,0,0,19,1,2,0,0,0,19,2,1,0,0,0
40 CL$:PRINTTAB(10,8)"OPZIONI:"TAB(10,11)
   "1□IMMISSIONE□DATI"TAB(10,13)"2□
   VISIONE/MODIFICA□DATI"TAB(10,15)"3□
```

```
   GRAFICO"TAB(10,17)"4□FINE LAVORO"
50 G=GET
60 IF G=49 THEN PROCINITVAL: GOTO 40
70 IF G=52 THEN 130
80 IF A=1 THEN 100
90 PRINT'"NON□SONO□STATI□IMMESSI□
   DATI.":FOR T=1 TO 2000:NEXT:GOTO 40
100 IF G=50 THEN PROCEDIT: GOTO 40
110 IF G=51 THEN PROCVIEW
120 GOTO 40
130 CLS
140 END
150 DEF PROCINITVAL
160 IF A=1 THEN PRINT'"SONO□GIÀ□
    STATI□IMMESSI□ALCUNI□DATI□":
    FOR T=1 TO 2000: NEXT: ENDPROC
170 A=1
180 CLS
190 INPUT'"NOME□DEL□GRAFICO□",T$:T$
    =LEFT$(T$,25)
200 INPUT'"NOME□PER□L'ASSE□X",X$:X$
    =LEFT$(X$,25)
210 INPUT'"NOME□PER□L'ASSE□Y",Y$:Y$
    =LEFT$(Y$,25)
220 INPUT'"QUANTE□BARRE",N
230 IF N=0 THEN 220
240 CLS
250 PRINT'"ADESSO□IMMETTERE□I□
    DATI"'
260 PRINTTAB(12)"BARRA","VALORE"
270 DX = 1000/N
280 DIM Y(N)
290 FOR T = 1 TO N
300 PRINTTAB(12)"N.□□";T"□□□";:INPUT
    ""Y(T)
310 NEXT
320 ENDPROC
330 DEF PROCVIEW
340 MAX=0:MIN=0
350 FOR T=1 TO N
360 IF Y(T)>MAX THEN MAX=Y(T)
370 IF Y(T)<MIN THEN MIN=Y(T)
380 NEXT
390 PROCSCALEPOS
400 PROCSCALENEG
410 PROCNORM
420 CLS
430 GCOL0,3
440 R=MAX3-MIN3
450 MOVE242,60:DRAW242,960
460 DY=900/R
470 YAX=60-MIN3/R*900
480 MOVE242,YAX:DRAW1200,YAX
490 W=1000/N-8
500 PROCYAXIS
510 PROCNAMES
520 FOR T=1 TO N
530 PROCBLOCK
540 NEXT
550 G=GET:ENDPROC
560 DEF PROCBLOCK
570 IF Y(T)>0 THEN GCOL0,1 ELSE
    GCOL0,2
580 MOVE250+(T-1)*DX,YAX:MOVE250+
    W+(T-1)*DX,YAX
590 PLOT85,250+(T-1)*DX,YAX+Y(T)*DY/
    FAC:PLOT85,W+(T-1)*DX+250,YAX+
    Y(T)*DY/FAC
600 ENDPROC
610 DEF PROCSCALEPOS
620 FAC=0
630 IF MAX=0 THEN 650
640 FAC = 10∧INT(LOG(MAX))
650 ENDPROC
660 DEF PROCSCALENEG
670 FAC2 = 0
680 IF MIN = 0 THEN 700
690 FAC2 = 10∧INT(LOG(-MIN))
700 ENDPROC
710 DEF PROCNORM
720 IF FAC2>FAC THEN FAC=FAC2
730 MAX3=INT(MAX/FAC+1):MIN3=INT
    (MIN/FAC-1)
740 IF MIN=0 THEN MIN3=0
750 IF MAX=0 THEN MAX3=0
760 ENDPROC
770 DEF PROCYAXIS
780 GCOL0,3
790 VDU5
800 FOR T=MIN3 TO MAX3
810 MOVE0,70+(T-MIN3)*DY:PRINTTAB
    (7-LEN(STR$(T)));T
820 NEXT
830 VDU4
840 ENDPROC
850 DEF PROCNAMES
860 GCOL0,3
870 VDU5
880 FOR T=1 TO LEN(Y$)
890 MOVE64,900-T*32:PRINTMID$(Y$,T,1)
900 NEXT
910 MOVE320,31:PRINTX$
920 VDU4
930 VDU30:PRINT"X";FAC;TAB(15)T$
940 ENDPROC
950 DEF PROCEDIT
960 VDU12:PRINT"PREMERE□UNO□SPAZIO
    □PER□MODIFICARE□IL□DATO□
    PRESENTATO□OPPURE□RETURN□PER
    □PASSARE□AL□SUCCESSIVO"
970 PRINT'
980 PRINT"□□□□BARRA□SENZA□□□
    □□□□VALORE"
990 FOR T=1 TO N
1000 PRINTT,Y(T)
1010 IF GET<>32 THEN 1030
1020 PRINTT;"□",;CHR$(8);:INPUT""Y(T)
1030 NEXT
1040 ENDPROC
```


T

```
10 PMODE 4,1
20 CLS
30 PRINT@45,"menu":PRINT@102,"1-□
   IMMISSIONE□DATI":PRINT@166,"2-□
   GRAFICO□A□BARRE":PRINT@230,"3-
   □VISIONE/MODIFICA□DATI":PRINT@294,
   "4-□FINE□LAVORO"
40 A$=INKEY$:IF A$<"1" OR A$>"4"
   THEN 40
50 IF A$="1" AND DA=1 THEN 80
60 ON VAL(A$) GOSUB 1000,2000,3000,4000
70 GOTO 20
80 PRINT@484,"SEI□SICURO□(S/N)□?";
90 A$=INKEY$:IF A$<>"S" AND A$<>
   "N" THEN 90
100 IF A$="S" THEN CLEAR200:A$="1":
    GOTO 60
110 GOTO 20
1000 DA=1:CLS:INPUT"□NOME□PER□
     L'ASSE□X";X$
1010 X$=LEFT$(X$,32):IF X$="" THEN X$
     ="ASSE□X"
1020 MD=INT(16-LEN(X$)/2)
1030 INPUT"□NOME□PER□L'ASSE□Y";Y$
1040 Y$=LEFT$(Y$,12):IF Y$="" THEN Y$
     ="ASSE□Y"
1050 HT=INT(6-LEN(Y$)/2)
1060 INPUT"N.□DI□BARRE□";NB
1070 NB=INT(NB):IF NB<1 THEN 1060
1080 BL=INT(26/NB):IF BL<1 THEN BL
     =1
1090 DIM A(NB)
```

```
1100 PRINT
1110 FOR K=1 TO NB
1120 PRINT"□VALORE□DELLA□BARRA";K;:
   INPUT A(K)
1130 NEXT
1140 RETURN
2000 IF DA=0 THEN PRINT@455,"dati□
   non□immessi":FOR K=1 TO 2000:NEXT:
   RETURN
2010 TP=0:BT=0
2020 FOR K=1 TO NB
2030 IF A(K)>TP THEN TP=A(K)
2040 IF A(K)<BT THEN BT=A(K)
2050 NEXT
2060 IF TP=0 AND BT=0 THEN PRINT
   "TUTTI□I□VALORI□A□ZERO":FOR K=
   1 TO 2000:NEXT:RETURN
2070 CLS:PRINT@64,"GRAFICO□CON□
   SCALA□(S/N)?"
2080 A$=INKEY$:IF A$<>"S" AND A$<
   >"N" THEN 2080
2090 IF A$="N" THEN 2120
2100 IF TP>0 THEN E=INT(LOG(TP)/LOG
   (10)):TP=INT (1+TP/(10↑E))*10↑E
2110 IF BT<0 THEN E=INT(LOG(-BT)/
   LOG(10)):BT=-INT(1-BT/(10↑E))*10↑E
2120 IN=178/(TP-BT)
2130 ST=INT(IN*TP)
2140 T$="":IF TP=0 THEN 2160
2150 IF TP>ABS(BT) THEN E=2*INT(LOG
   (TP)/LOG(1000)) ELSE E=2*INT(LOG(ABS
   (BT))/LOG(1000))
2160 T$="+"+MID$(STR$(INT(.5+TP)/10
   ↑E)),2):B$="":IF BT=0 THEN 2180
2170 B$ = STR$ (INT(.5+BT/10↑E))
2180 SL=1:LP=NB:IF NB>26 THEN LP
   =26
2190 POKE 179,243:PCLS:POKE 65475,0:
   POKE 65477,0:POKE 65479,0
2200 POKE 179,2
2210 LINE(40,0)-(47,191),PRESET, BF
2220 IF T$="" THEN 2260
2230 FOR K=1 TO LEN(T$):P=ASC(MID$
   (T$,K,1))AND63:FOR M=0 TO 11
2240 POKE 1540-LEN(T$)+32*M+K,P
2250 NEXT M,K
2260 IF B$="" THEN 2300
2270 FOR K=1 TO LEN(B$):P=ASC(MID$
   (B$,K,1))AND63:FOR M=0 TO 11
2280 POKE 6916-LEN(B$)+M*32+K,P
2290 NEXT M,K
2300 FOR K=1 TO LEN(X$)
2310 P=ASC(MID$(X$,K,1))AND63
2320 FOR M=182 TO 190:POKE 1503+MD
   +32*M+K,P:NEXT
2330 NEXT
2340 FOR K=1 TO LEN(Y$)
2350 FOR M=0 TO 11
2360 P=ASC(MID$(Y$,K,1))AND63
2370 POKE 1568+384*(K+HT)+32*M,P
2380 NEXT M,K
2390 FOR K=5 TO 31
2400 POKE 1536+32*ST+K,128
2410 NEXT
2420 FOR K=ST TO 0 STEP -22
2430 POKE 1541+K*32,202
2440 NEXT
2450 FOR K=ST TO 178 STEP 22
2460 POKE 1541+K*32,202
2470 NEXT
2480 FOR K=SL TO LP:CO=(CO+1)AND1
2490 POKE 178,35+CO*64:POKE 179,3+
   CO*192
2500 IF INT(A(K)*IN)>0 THEN LINE(48+8*
   (K-SL)*BL,ST-1)-(47+8*(K-SL+
   1)*BL,ST-INT(A(K)*IN)),PSET,BF
2510 IF FIX(A(K)*IN)<0 THEN LINE(48+8*
   (K-SL)*BL,ST)-(47+8*(K-SL+1)*
   BL,ST-FIX(A(K)*IN)),PRESET,BF
2520 NEXT K
2530 IF LP=NB THEN 2570
2540 SL=LP+1:LP=LP+26:IFLP>
   NB THEN LP=NB
2550 IF INKEY$="" THEN 2550
2560 POKE 179,243:LINE(48,0)-(255,178),
   PRESET,BF:GOTO 2390
2570 IF INKEY$="" THEN 2570
2580 RETURN
3000 IF DA=0 THEN PRINT@455,"nessun
   □dato□immesso":FOR K=1 TO 2000:
   NEXT:RETURN
3010 SL=1:LP=NB:IF NB>14 THEN LP
   =14
3020 CLS:PRINT"□barra","valore"
3030 FOR K=SL TO LP
3040 PRINTK,A(K)
3050 NEXT
3060 IF NB=LP THEN 3130
3070 PRINT@480,"v□PER□VARIARE,□
   QUALSIASI□TASTO□PER□
   CONTINUARE";
3080 A$=INKEY$:IF A$="" THEN 3080
3090 IF A$= "V" THEN 3170
3100 SL=LP+1:LP=LP+14
3110 IF LP>NB THEN LP=NB
3120 GOTO 3020
3130 PRINT@480,"v□PER□VARIARE,□
   QUALSIASI□TASTO□PER□
   CONTINUARE";
3140 A$=INKEY$:IF A$="" THEN 3140
3150 IF A$="V" THEN 3170
3160 RETURN
3170 CLS:PRINT:INPUT"□NUMERO□DEL□
   DATO□DA□VARIARE□?";E
3180 E = INT(E):IF E<1 OR E>NB THEN
   3170
3190 PRINT:PRINT"NUOVO□VALORE□DEL□
   DATO";E;:INPUT A(E)
3200 GOTO 3020
4000 CLS:PRINT@37,"SEI□SICURO□(S/N)□
   ?"
4010 A$ = INKEY$:IF A$<>"S" AND A$<
   >"N" THEN 4010
4020 IF A$ = "N" THEN RETURN
```

LO SVILUPPO DI UN'AVVENTURA

Eccoci in un mondo di fantasia, creato da noi stessi per stupire e anche un po' per sconvolgere gli amici. Diamo uno sguardo al mondo e alla storia dei giochi di avventura

Per chi desideri un po' di tregua dai giochi a base di velocità e mira, l'alternativa è il gioco d'avventura, un genere di gioco grazie al quale ci si immerge totalmente in un mondo di fantasia creato dallo stesso programmatore. Esercitando intelligenza, saggezza e conoscenza di fatti ed eventi insoliti, si conduce un viaggio alla ricerca dell'obiettivo creato dall'ideatore.

Nelle successive sezioni di Giochi al Computer, verrà spiegato come creare da zero un proprio gioco d'avventura. Per adesso, ecco un'introduzione a questo interessante genere.

LA STORIA DELLE "ADVENTURE"

L'idea di scrivere giochi d'avventura proviene dalla popolarità di giochi classici (non da computer) quali il "Dungeons e Dragons" e dal desiderio di fare con il computer qualcosa di più interessante della semplice elaborazione dati. In "Dungeons e Dragons" i giocatori entrano nei panni di alcuni personaggi e si affrontano l'un l'altro in un mondo, Dungeon, creato dal Signore di Dungeon. Questo ruolo, nei giochi d'avventura computerizzati, è in parte svolto del programmatore che, come il Signore di Dungeon, crea un proprio mondo. Il giocatore, d'altra parte, assume un ruolo simile a quello di uno dei personaggi.

Tuttavia, nei giochi d'avventura al computer, i giocatori di solito non possono scegliere il tipo di personaggio: è il computer a fare questa scelta. Nelle versioni più sofisticate, si può anche scegliere l'equipaggiamento da portarsi appresso, prima di partire verso l'obiettivo, ma partecipazione e verosimiglianza dell'avventura dipendono tutte dall'ideatore. La prima avventura fu scritta in FORTRAN, anziché in BASIC, su una grande unità di elaborazione. Il programma occupava più di 300 Kbyte di memoria, molto di più di quella di un microcomputer.

La storia vera e propria cominciò però nel 1978, quando Scott Adams trasferì alcune idee su un micro TRS 80, dimostrando la possibilità di scrivere un'avventura soddisfacente in uno spazio di memoria molto minore.

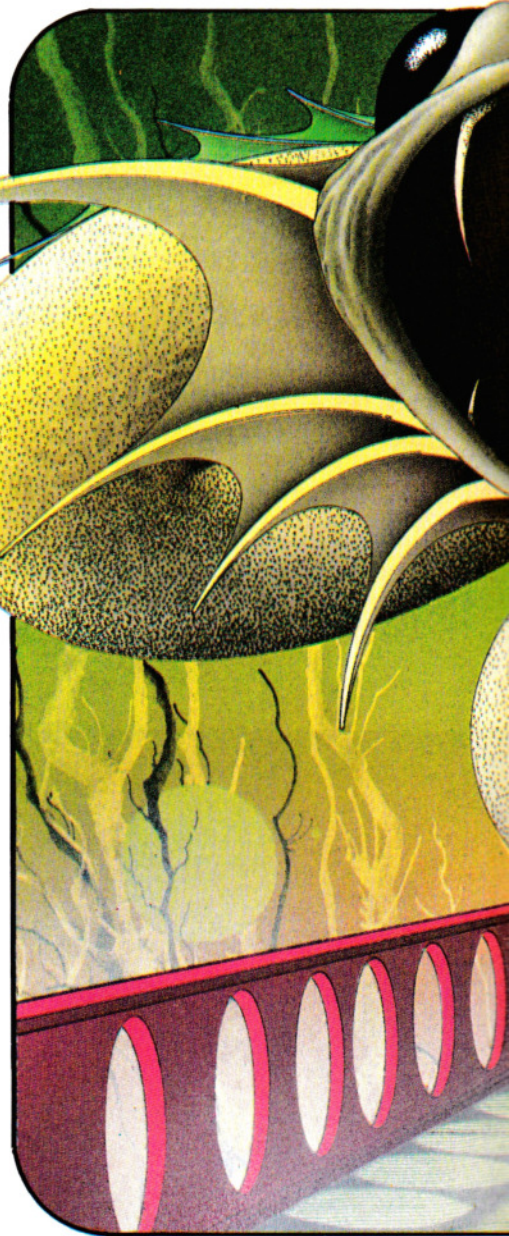
I temi adottati da Adams per i suoi giochi (Adventureland, Covo dei Pirati, La casa dei misteri e Il Conte), sono stati da allora una fonte inesauribile di ispirazione per gli ideatori di giochi.

TIPI DI AVVENTURA

Nel gioco originale, ma anche in quelli di Adams, sullo schermo vengono visualizzati soltanto dei testi. Questo tipo di gioco, esclusivamente testuale, è ancora molto apprezzato e, sostengono alcuni, costituisce la vera avventura.

In effetti, nelle avventure puramente testuali, la trama esiste soltanto nella mente del giocatore che, se il programma è buono, sarà del tutto preso dallo svolgersi degli eventi.

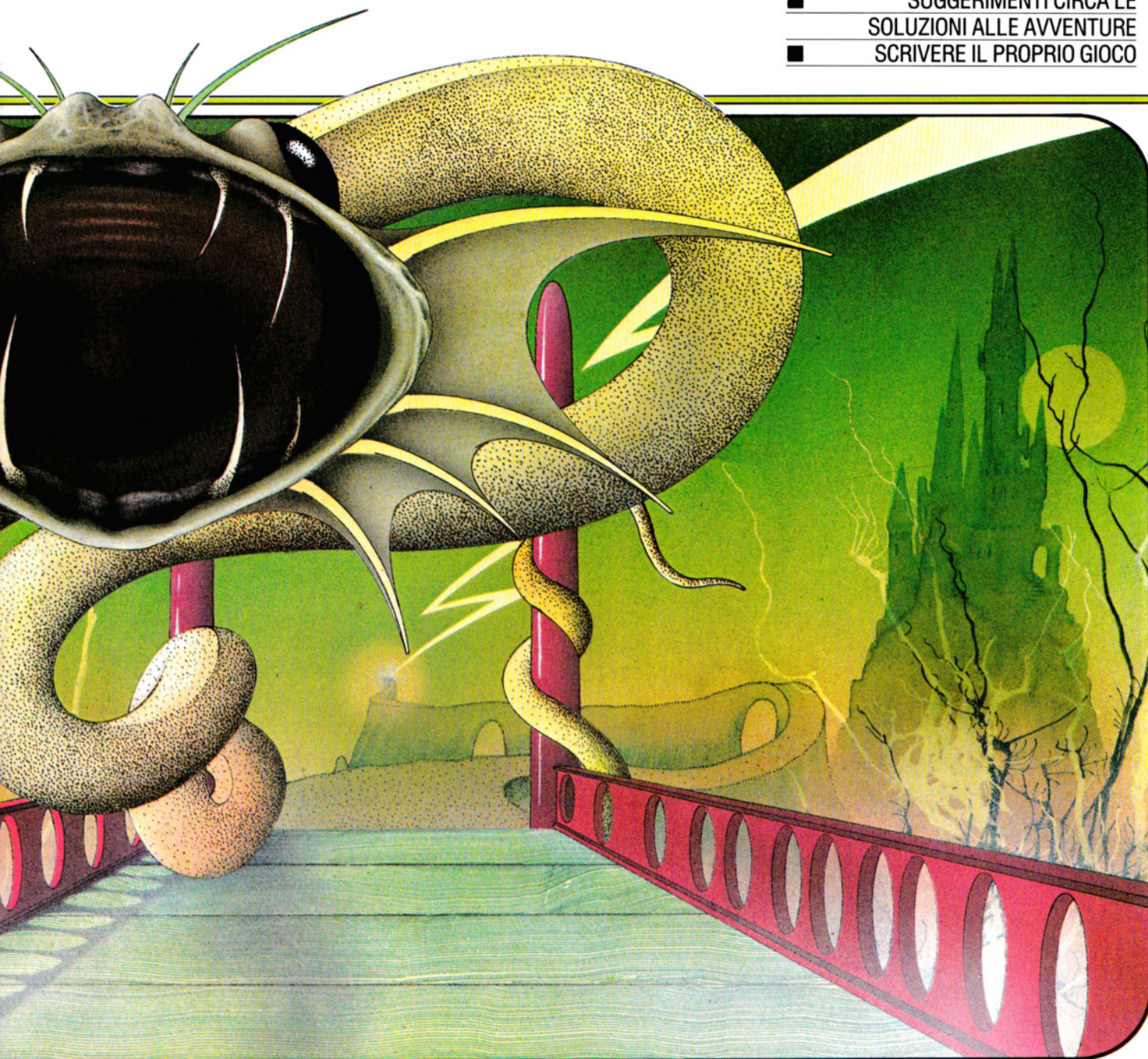
Solo una grafica molto sofisticata può competere con la nostra immaginazione. Per esempio, si può immaginare un orco molto più feroce di quanto si riesca a disegnare con la migliore grafica.

Una considerazione più concreta contro l'uso della grafica è che la gran quantità di memoria ad essa dedicata potrebbe invece servire per allargare la portata del gioco. Inoltre, il ritmo dell'avventura si infiacchisce, se il giocatore deve attendere ogni volta il formarsi del disegno.

Certe avventure assegnano un punteggio al raggiungimento di determinate fasi del gioco, in modo da sapere, a qualsiasi livello si venga eventualmente uccisi, come ci si è comportati. Altri, invece, esprimono un giudizio del tipo 'buffone o esperto'. Un tipo di avventura diametralmente opposto non fornisce intenzionalmente alcuna indicazione su come si stia procedendo, né su quanto si è vicini alla

Cosa fare se si resta bloccati in mezzo a un'avventura?

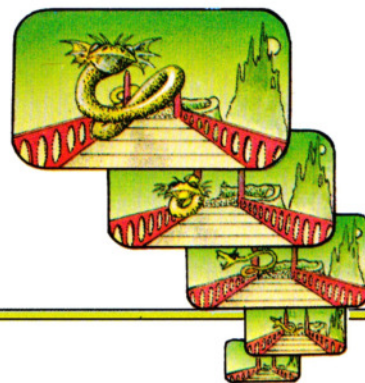
Prima di arrenderci, riproviamo dopo un po' di tempo: può venire in mente un'altra idea o strategia. Ma se si è proprio bloccati, le avventure in commercio forniscono spesso fascicoletti con le soluzioni, o danno modo di richiederli per posta.

Un altro canale da seguire per un aiuto sono le rubriche di alcune riviste specializzate.

- COS'È UN GIOCO D'AVVENTURA
- COME SI GIOCA
- SUGGERIMENTI CIRCA LE SOLUZIONI ALLE AVVENTURE
- SCRIVERE IL PROPRIO GIOCO

meta: la soddisfazione finale proviene dalla risoluzione di una lunga serie di rompicapi, che ci avvicinano gradualmente al termine della ricerca e all'obiettivo finale.

COME GIOCARE

Di norma, quando si inizia un'avventura, viene fornita una descrizione sommaria sul mondo in cui ci si addentra: se cioè si tratta di un luogo inesplorato o di un pianeta sperduto oppure di un mondo del tutto fantastico.

Il gioco può ambientarsi nel passato, nel presente o nel futuro... o anche in tutti e tre i momenti. In genere, viene data qualche indicazione iniziale (chi comanda nel mondo, qual'è il ruolo del giocatore, chi sono gli amici e chi i nemici e via dicendo) e, dato più importante, cosa occorre fare per riuscire nell'avventura e vin-

LA SOLUZIONE DELLE AVVENTURE

Normalmente, esiste una sola soluzione per un'avventura; ad esempio: raccogliere tutto il tesoro e riportarlo al sicuro, o eliminare il rappresentante del Male e sfuggire illesi.

Il resto è una sequenza fissa di problemi da risolvere. Le probabilità sono tali che i tentativi saranno molti prima che l'avventura finisca. Infatti, le avventure più belle sono quelle che richiedono giorni, settimane, o addirittura mesi di fatica! Ci sono però alcune regole e suggerimenti per aiutare a risolvere quasi tutte le avventure in meno tempo.

Quasi senza eccezioni, gli oggetti che si trovano nelle avventure servono a qualcosa. Sarebbe uno spreco di memoria se il programmatore lasciasse in giro troppi diversivi, anche se alcuni oggetti saranno armi a doppio taglio. Per esempio, può essere necessario portarsi dietro un sacco di monete d'oro per pagare il pedaggio sul ponte, ma se si deve invece passare il fiume a nuoto, il peso ci farà annegare. In genere, conviene prendere quanti più oggetti possibile, ma in qualche occasione si vedrà che non si possono tenere tutti. Molti di essi servono solo una volta nell'avventura, salvo forse una spada, ad esempio, che servirà più volte per eliminare i cattivi. Se c'è un limite al numero di oggetti da portare con sé, ricordiamo che spesso è prudente scartare gli oggetti dopo l'uso.

È *sempre* bene tracciare una mappa, segnandovi i nomi delle stanze e le cose interessanti in ognuna, oltre agli oggetti che vi si trovano, le entrate, le uscite e le loro direzioni.

Questa mappa ci risparmierà un sacco di tempo, evitando di tornare sui propri passi, cosa che accade spesso durante il gioco. Se ci sono oggetti da abbandonare, perché non si può portare tutto con sé, si segni la loro posizione sulla mappa. E, altrettanto importante, una mappa dà la garanzia di esplorare tutta l'avventura senza per questo ricadere per l'ennesima volta nelle "solite" sabbie mobili.

La maggior parte dei giochi prevede la visualizzazione di un inventario degli oggetti che si hanno con sé e per esaminarlo di fronte a un ostacolo basta scrivere INVENTARIO, INVE o solo I, a seconda dell'avventura.

Allo stesso modo, alcune avventure permettono di chiedere aiuto e la formula dipende anche stavola dal gioco. Può esserci o non esserci un aiuto, ma alla peggio ci si troverà di fronte a "NESSUN AIUTO QUI!". Alcuni giochi seguono da vicino la trama di un particolare libro e in questo caso studiarsi il libro in questione è senz'altro una buona idea. A volte sono state riprese solo piccole sezioni o idee e, se ci sembra di riconoscere qualcosa, proviamo a ricercarla sul libro.

Analogamente, se un personaggio ripugnante che brandisce una pesante mannaia ci sbarra la strada chiedendoci quale sia il diametro della Terra, non cerchiamo di tirare a indovinare, ma corriamo a consultare un'enciclopedia!

Un'altra trappola molto usata sono i quiz con giochi di parole: attenzione, non sempre sono facili come sembra. Teniamoci quindi vicino un vocabolario o un dizionario dei sinonimi per risolvere doppi sensi e frasi ambigue: il programmatore potrebbe aver introdotto LUSTRARE, ma non STROFINARE, ad esempio.

Un ultimo avvertimento: se l'avventura che si sta giocando permette di memorizzare su nastro o su disco una particolare fase del gioco, per poterlo giocare a più riprese, è bene sfruttare l'occasione prima di affrontare una difficoltà. Se andasse male, si può sempre riprende-

re da dove si era rimasti e tentare così più volte l'assalto a un drago o il passaggio di un ponte pericolante.

SCRIVERE AVVENTURE

Scrivere avventure è un buon modo per esercitarsi in BASIC: vengono utilizzati quasi tutti gli aspetti importanti del linguaggio, dall'uso delle variabili e della segmentazione delle stringhe, ai vari tipi di PRINT per la formattazione dello schermo, alle matrici ecc. L'unica barriera nella produzione di giochi d'alta qualità è la propria immaginazione.

Prima di mettersi a programmare un'avventura, occorre però avere idee ben chiare su cosa si vuol fare e sul soggetto. Trama, ostacoli, trappole e così via devono essere tutti predefiniti per risparmiarsi possibili problemi.

Prima di tutto, buttiamo giù le idee su un foglio: anche se non si ha un quadrao completo di tutte le implicazioni, basta un'idea generale, un'ambientazione e qualche ostacolo sul cammino del giocatore. Nelle prossime lezioni verrà spiegato come tramutare l'idea di un gioco in un programma, imparando a inserire le proprie idee originali. Si faccia molta attenzione al mondo che si sceglie per ambientare l'avventura e si cerchi di renderlo il più interessante possibile: ci vuole un bel po' di fatica per scrivere un'avventura divertente ambientata in un grigio ambiente d'ufficio!

Ci si può ispirare a libri, romanzi, film, commedie, oppure a idee totalmente nuove. Si può prendere a prestito qualcosa da altre storie, anche se la fonte migliore è... un buon incubo! Comunque sia, occorre sempre un tema centrale da sviluppare nel corso dell'avventura.

Si cerchi sempre un giusto equilibrio tra sfida e impossibilità: non val la pena dedicare un sacco di tempo a scrivere un'avventura se poi chiunque riesce a risolverla nel giro di mezz'ora. Viceversa, non ci faremo certo degli amici tra i giocatori, se le nostre storie sono del tutto impossibili da risolvere.

La regola è *dare a chi punta una possibilità, ma non troppe!*

Si cerchi anche di non lasciare troppi ambienti vuoti, che nulla aggiungono all'avventura, mentre occupano spazio vitale in memoria (per non parlare del fastidio che danno!).

Non si parta con storie troppo complicate perché i problemi da affrontare sono complessi finché non si è acquisita buona pratica.

Si valutino bene le conseguenze di ogni tentativo ambizioso e si tenga d'occhio anche la quantità di memoria che rimane alla macchina.

Nell'avventura che viene sviluppata nel corso di Giochi al Computer, troviamo molte frasi REM. Per risparmiare memoria, in un'avventura lunga conviene ometterle del tutto, anche se ciò comporta una più difficile comprensione del listato.

Altro spazio su cui economizzare memoria è quello delle descrizioni degli ambienti (senza peraltro accorciarle troppo, altrimenti l'avventura perde di fascino). Nella prossima lezione vedremo come passare dall'idea dell'avventura a una mappa e impostare il programma.

Quanta memoria serve per scrivere un gioco d'avventura?

La risposta giusta sarebbe: "tutta quella possibile"! L'avventura sviluppata nel corso di questa sezione è molto semplice e occupa circa 5K di BASIC, ma si noterà che non è a un livello confrontabile con quanto reperibile in commercio. Le avventure migliori hanno un gran numero di stanze e di indovinelli e, inoltre, risultano difficili per l'accumularsi dei problemi. Con una quantità di memoria limitata si è costretti a selezionare un numero di problemi minori, ma di maggior complessità, per impedire una soluzione rapida e di scarsa soddisfazione. Un computer con 16K di memoria può considerarsi il minimo.

QUANTA MEMORIA RESTA?

Nello scrivere un lungo gioco d'avventura è facile scoprire che si stanno forzando i limiti di memoria della macchina. È ovvio che il problema è più acuto quanto minore è la memoria della macchina e, del resto, è una perdita di tempo scrivere un avventura per un computer con meno di 16K.

Per ogni macchina esiste un modo di controllare quanta memoria resta. Basta una semplice routine:

Sullo Spectrum, si scriva:

```
Print (PEEK 23730 + 256*PEEK 23731)—
    (PEEK 23653 + 256*PEEK 23654)
```

Così si mette in conto sia il programma in sé che lo spazio delle variabili. Una migliore valutazione della memoria restante si ha appena dopo aver eseguito un RUN del programma (se ciò è possibile nel corso della stesura).

Sui Commodore, si ottiene la memoria restante digitando:

```
PRINT FRE (0)
```

La memoria restante tiene conto sia dello spazio occupato dal programma che dalle variabili, ma in tal caso occorre usare la linea appena descritta dopo aver dato un RUN del programma, se ciò è possibile nel corso della stesura.

Sugli Acorn, si ottiene la memoria restante digitando:

```
PRINT HIMEM — TOP
```

Questo numero comprende solo lo spazio occupato dal programma: lo spazio delle variabili non è valutabile, e di ciò va tenuto conto lasciando spazio sufficiente. Per ottenere l'ammontare massimo di memoria si usi il Mode 7 sul BBC e il Mode 6 sull'Electron.

Sul Dragon e sul Tandy, si trova la memoria restante digitando:

```
PRINT MEM
```

Il risultato tiene conto dello spazio occupato dal programma e dalle variabili, perciò conviene eseguire un RUN nel corso della stesura, se ciò è possibile, per avere un'indicazione reale sulla memoria totale impiegata.

Esiste un metodo molto facile per aumentare la quantità di memoria disponibile sul Dragon per i programmi in BASIC. Prima di cominciare a programmare, si digitino queste due POKE:

```
POKE 25,6
POKE 26,1
NEW
```

Si guadagnano così altri 6Kbyte per il programma, facendo utilizzare al programma BASIC lo spazio normalmente riservato alla grafica ad alta risoluzione, ossia quattro pagine di 1 Kbyte e mezzo.

Se si usano le POKE, ci si accerti di non usare comandi grafici nell'avventura, altrimenti si sciupa il programma. Quando, dopo aver memorizzato il programma su nastro o su disco, lo si vuole ricaricare in memoria, non ci si dimentichi di digitare prima le POKE, seguite da un NEW.

TIRO INCROCIATO CON LE MATRICI

- PREDISPORRE UNA MATRICE
- L'IMMISSIONE DEI DATI
- SCRIVERE I PROGRAMMI
- ESTRARRE INFORMAZIONI
- ADATTARE LA MATRICE

Se si hanno molti dati interrelati, conviene assegnarli a una matrice, anziché disperderli in più variabili. In questo modo il computer può ripescarli con facilità

L'uso di matrici per contenere informazioni all'interno di un programma è stato in parte illustrato a pagina 152. Le matrici, in BASIC, sono uno strumento davvero efficace, ma quelle *a due dimensioni* si dimostrano spesso ancora più utili, per archiviare molti dati *interrelati*.

Si può paragonare una matrice a due dimensioni (o *bidimensionale*) a una cassettiera da muro, composta da diverse file di piccoli scomparti, ognuno dei quali contiene un elemento d'informazione. È un modo comodo per archiviare dati, poiché consente di accedere a un qualsiasi elemento, semplicemente indicandone la riga e la colonna.

Riferendosi alla **figura 1**, la "matrice" cassettiera può contenere veri articoli (viti, in questo caso): numerando le righe e le colonne è facile individuare lo scomparto che contiene le viti di ottone da 38 mm, ad esempio. In questo caso il dato nella matrice è costituito dal numero delle viti in ogni scatola. Le matrici del BASIC, però, contengono informazioni astratte, come ad esempio il numero dei concessionari d'auto per ogni marca in ogni nazione.

Informazioni di questo genere sono spesso raccolte come risultato di un'indagine ed è questa la base del programma qui esaminato. Comunque, la struttura del programma è la stessa per ogni matrice a due dimensioni, a prescindere dal tipo di informazione contenuta.

A titolo d'esempio, supponiamo che un insegnante faccia un'indagine sugli animali posseduti da un gruppo di bambini. Prima occorre stilare un elenco dei bambini e poi, a fianco di ciascun nome, il numero di animali di ciascun tipo posseduto.

L'elenco potrebbe essere questo:

Marta: 2 canarini, 1 coniglio;
Guido: 1 cane, 4 pesciolini;
Luisa: 2 criceti, 1 gatto;
Gigi: 1 cane, 1 gatto, 1 criceto;
Roberto: 1 pulcino, 1 coniglio,
2 pesciolini.

Tuttavia, in una lista di questo tipo, è piuttosto difficile estrarre informazioni del tipo "quanti possiedono gatti" o "chi ha più di quattro animali". Se l'elenco si allunga, la difficoltà cresce ancor più.

Un metodo di gran lunga migliore è scrivere le informazioni sotto forma di tabella o di grafico, riportando i nomi degli

1. Le matrici sono utili per smistare piccoli articoli come queste viti

animali lungo un lato e quelli dei bambini lungo l'altro. Ma così facendo, senza rendercene conto, abbiamo predisposto una matrice a due dimensioni!

Con una quantità di dati piccola come questa, semplici statistiche si possono fare a mente, ma se i dati sono tanti, un computer diventa indispensabile. Se l'indagine, anziché a una sola classe si riferisse all'intera scuola, ci vorrebbe un bel po' di tempo per controllare manualmente chi abbia almeno un gatto o i nomi di quelli che hanno criceti. Il computer ci mette pochi secondi.

PREDISPORRE LA MATRICE

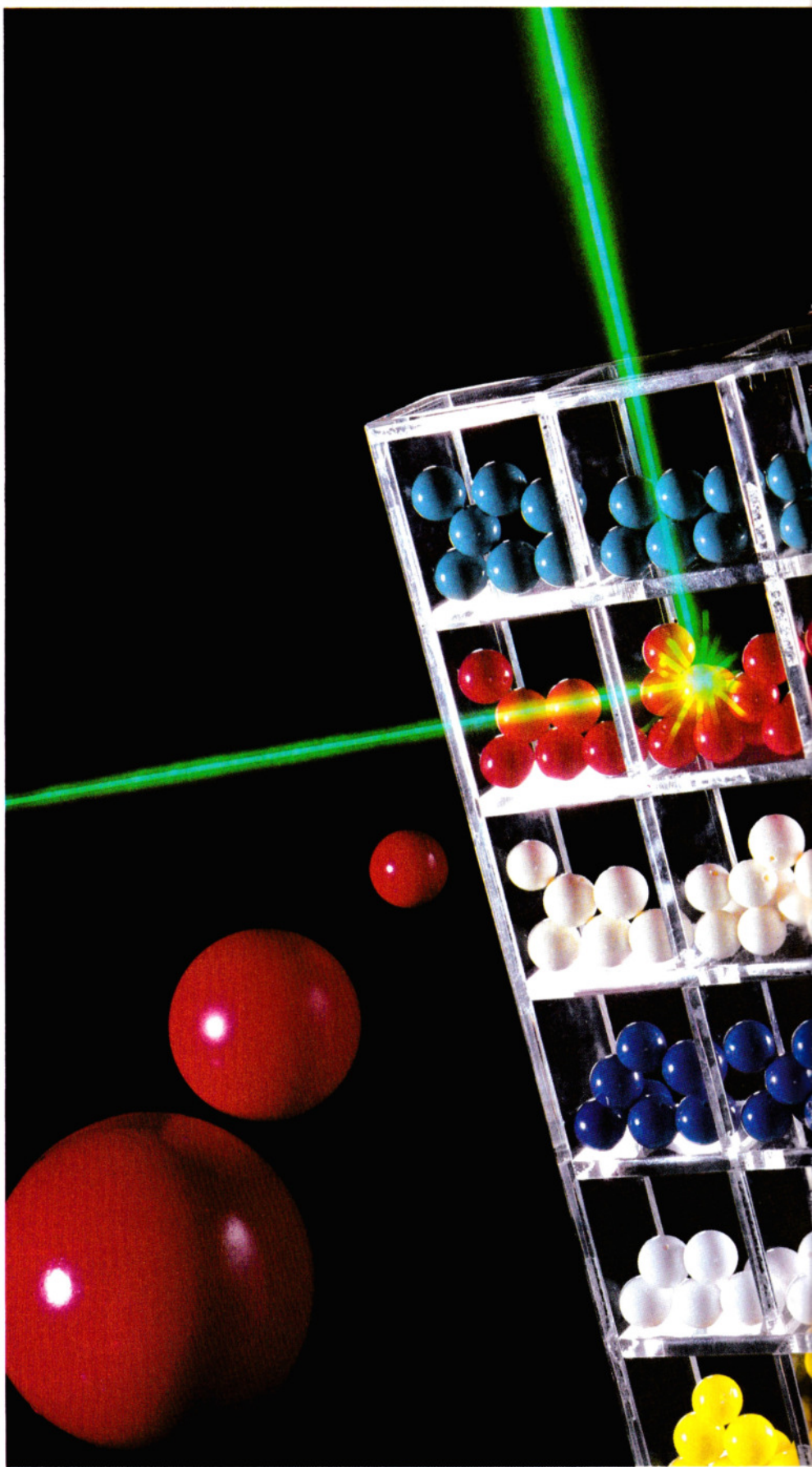
Il passo successivo consiste nell'immettere le informazioni nel computer. Alle pagine 152-155, si è visto come predisporre e usare una semplice matrice a una dimensione. Le matrici a due dimensioni sono molto simili e comportano solo un po' più di tempo per la preparazione.

Una matrice semplice può essere scritta nella forma A(N), dove N è il numero di elementi che la compongono; una matrice a due dimensioni si indica con A(R,C), dove R è il numero delle righe e C quello delle colonne. In effetti, potremmo anche scrivere A(C,R), con le colonne prima delle righe: basta essere coerenti e, una volta adottata una notazione, attenervisi.

Adesso, pur disponendo di una matrice a due dimensioni, occorre definire due matrici semplici per contenere le intestazioni delle colonne e delle righe: nel nostro caso, una matrice contiene i nomi dei bambini e un'altra i nomi degli animali. La matrice a due dimensioni serve a contenere i dati.

Indichiamo le matrici semplici con PET$(C) e CH$(R) e la matrice dei dati con N(R,C). Un'ultima matrice, PT(R), contiene il numero totale di animali per ogni riga.

Sullo Spectrum le matrici sono indicate, rispettivamente, con p$(c), c$(r), n(r,c) e p(r), dato che sono ammessi soltanto nomi di una sola lettera. Il programma non gira sullo ZX81, per la difficoltà di immettere tutti i dati. Ecco il programma per DIMensionare le matrici e leggere le DATA:

```
100 C=7: R=5
110 DIM PET$(C), CH$(R), N(R,C), PT(R)
120 FOR J=1 TO C: READ PET$(J): NEXT
130 FOR J=1 TO R: READ CH$(J): NEXT
140 FOR J=1 TO R
150 FOR K=1 TO C
160 READ N(J,K): NEXT K: NEXT J
3000 DATA CANARINO,GATTO,CANE,
   PESCIOLINO,CRICETO,CONIGLIO,PULCINO
```

2. La disposizione più conveniente, per queste palline, è in righe secondo il colore e in colonne secondo la dimensione

```
3010 DATA MARTA,GUIDO,LUISA,GIGI,
   ROBERTO
3020 DATA2,0,0,0,0,0,1
3030 DATA0,0,1,4,0,0,0
3040 DATA0,1,0,0,0,2,0
3050 DATA0,1,1,0,0,1,0
3060 DATA0,0,0,2,1,0,1
```

```
100 LET c=7: LET r=5
110 DIM p$(c,7): DIM c$(r,5): DIM n(r,c): DIM
   p(r)
120 FOR j=1 TO c: READ p$(j): NEXT j
130 FOR j=1 TO r: READ c$(j): NEXTj
140 FOR j=1 TO r
150 FOR k=1 TO c
160 READ n(j,k): NEXT k: NEXT j
3000 DATA "CANARINO", "GATTO", "CANE",
   "PESCIOLINO", "CRICETO", "CONIGLIO",
   "PULCINO"
3010 DATA "MARTA", "GUIDO", "LUISA",
   "GIGI", "ROBERTO"
3020 DATA 2,0,0,0,0,0,1
3030 DATA 0,0,1,4,0,0,0
3040 DATA 0,1,0,0,0,2,0
3050 DATA 0,1,1,0,0,1,0
3060 DATA 0,0,0,2,1,0,1
```

Per facilitare l'adattamento del programma (non è detto che altre applicazioni abbiano il medesimo numero di elementi), sono state usate le variabili R e C. Infatti, basta cambiare i valori alla linea 100 e, naturalmente, le informazioni nelle frasi DATA.

Si noti la sistemazione delle informazioni nelle DATA: una prima frase (linea 3000) per le intestazioni delle colonne, una seconda per le intestazioni delle righe, seguono quindi tante frasi quante sono le righe della matrice centrale. Si noti che occorre scrivere un dato per ogni elemento della matrice, anche se esso vale zero, altrimenti il BASIC segnala un errore del tipo "OUT OF DATA" (dati insufficienti).

Ora che i dati sono inseriti nel computer, va deciso cosa farne. Si può prevedere ogni sorta di opzioni possibili, tanto è il computer a fare tutto il lavoro.

IL MENU

La miglior forma per presentare le varie opzioni consiste in un menu, che, nel caso dell'indagine sugli animali, potrebbe essere il seguente:

```
300 LET found=0: CLS
310 PRINT" "MENU"
320 PRINT' "1 ELENCO ANIMALI"
330 PRINT' "2 ELENCO NOMI"
340 PRINT "3 IMMISSIONE NOMI (ANIMALI)"
350 PRINT "4 IMMISSIONE NOMI (BAMBINI)"
360 PRINT "5 IMMISSIONE NUMERO DI
   ANIMALI"
370 PRINT "6 VISIONE MATRICE"
380 PRINT" "QUALE OPZIONE"
390 INPUT a
395 IF a<1 OR a > 6 THEN GOTO 390
400 CLS
410 GOSUB (a*100+400)
420 GOTO 300
```

```
300 FOUND=0:PRINT "♥"
310 PRINT "MENU■"
320 PRINT "1 ELENCO ANIMALI"
330 PRINT "2 ELENCO NOMI"
340 PRINT "3 IMMISSIONE NOMI (ANIMALI)"
350 PRINT "4 IMMISSIONE NOMI (BAMBINI)"
360 PRINT "5 IMMISSIONE NUMERO DI
   ANIMALI"
370 PRINT "6 VISIONE MATRICE"
380 PRINT "■QUALE OPZIONE"
390 INPUT A
395 IF A<1 OR A>6 THEN GOTO 390
400 PRINT "♥"
410 ON A GOSUB 500,600,700,800,900,1000
420 GOTO 300
```

```
300 FOUND=0:CLS
310 PRINT""MENU"
320 PRINT' "1 ELENCO ANIMALI"
330 PRINT' "2 ELENCO NOMI"
340 PRINT "3 IMMISSIONE NOMI (ANIMALI)"
350 PRINT "4 IMMISSIONE NOMI (BAMBINI)"
360 PRINT "5 IMMISSIONE NUMERO DI
   ANIMALI"
370 PRINT "6 VISIONE MATRICE"
380 PRINT" "QUALE OPZIONE"
390 INPUT A
395 IF A<1 OR A>6 THEN 390
400 CLS
410 ON A GOSUB 500,600,700,800,900,1000
420 GOTO 300
```

```
300 FOUND=0:CLS
310 PRINT@45,"MENU"
320 PRINT@100,"1 ELENCO ANIMALI"
330 PRINT@132,"2 ELENCO NOMI"
340 PRINT@164,"3 IMMISSIONE NOMI
   (ANIMALI)"
350 PRINT@196,"4 IMMISSIONE NOMI
   (BAMBINI)"
360 PRINT@228,"5 IMMISSIONE NUMERO DI
   ANIMALI"
370 PRINT@260,"6 VISIONE MATRICE"
380 PRINT:PRINT"QUALE OPZIONE";
390 INPUT A
```

```
395 IF A<1 OR A>6 THEN 390
400 CLS
410 ON A GOSUB 500,600,700,800,900,1000
420 GOTO 300
```

SCRIVERE LE SUBROUTINE

Come base di partenza, quanto esposto finora può bastare: in seguito si possono aggiungere altre opzioni. Con la prima opzione si ottiene un semplice elenco degli animali, grazie a un ciclo che scorre la matrice PET$() o p$(). La seconda opzione agisce in modo analogo, ma per i nominativi dei bambini.

```
499 REM **OPZIONE 1**
500 PRINT'"ELENCO ANIMALI"
510 PRINT'
520 FOR j = 1 TO c
530 PRINT p$(j)
540 NEXT j
550 PRINT'"PREMERE UN TASTO PER
    TORNARE AL MENU"
560 PAUSE 0
570 RETURN
599 REM **OPZIONE 2**
600 PRINT'"ELENCO BAMBINI"
610 PRINT'
620 FOR j=1 TO r
630 PRINT c$(j)
640 NEXT j
650 PRINT'"PREMERE UN TASTO PER
    TORNARE AL MENU"
660 PAUSE 0
670 RETURN
```

```
499 REM # # OPZIONE 1 # #
500 PRINT "ELENCO ANIMALI"
510 PRINT
520 FOR J=1 TO C
530 PRINT PET$(J)
540 NEXT J
550 PRINT "▣PREMERE UN TASTO PER
    TORNARE AL MENU"
560 GET A$:IF A$="" THEN 560
570 RETURN
599 REM # # OPZIONE 2 # #
600 PRINT "ELENCO BAMBINI"
610 PRINT
620 FOR J=1 TO R
630 PRINT CH$(J)
640 NEXT J
650 PRINT "▣PREMERE UN TASTO PER
    TORNARE AL MENU"
660 GET A$:IF A$="" THEN 660
670 RETURN
```

```
499 REM **OPZIONE 1**
500 PRINT'"ELENCO ANIMALI"
510 PRINT'
520 FOR J=1 TO C
530 PRINT PET$(J)
540 NEXT
550 PRINT'"PREMERE UN TASTO PER
    TORNARE AL MENU"
560 A=GET
570 RETURN
599 REM **OPZIONE 2**
600 PRINT'"ELENCO BAMBINI"
610 PRINT'
620 FOR J=1 TO R
630 PRINT CH$(J)
640 NEXT
650 PRINT'"PREMERE UN TASTO PER
    TORNARE AL MENU"
660 A=GET
670 RETURN
```

```
499 REM **OPZIONE 1**
500 PRINT@99,"ELENCO ANIMALI"
510 PRINT:PRINT
520 FOR J=1 TO C
530 PRINT PET$(J)
540 NEXT
550 PRINT:PRINT"PREMERE UN TASTO PER
    TORNARE AL MENU"
560 IF INKEY$="" THEN 560
570 RETURN
599 REM **OPZIONE 2**
600 PRINT@99,"ELENCO BAMBINI"
610 PRINT:PRINT
620 FOR J=1 TO R
630 PRINT CH$(J)
640 NEXT
650 PRINT:PRINT"PREMERE UN TASTO PER
    TORNARE AL MENU"
660 IF INKEY$="" THEN 660
670 RETURN
```

La terza opzione è più interessante: digitando il nome di un animale, il computer visualizza il nominativo di chi ne possiede almeno uno, come pure il numero di animali posseduti:

```
699 REM **OPZIONE 3**
700 PRINT'"IMMETTERE NOME
    DELL'ANIMALE"
705 DIM i$(7): INPUT LINE i$
710 PRINT'"CHI HA UN□"; i$
720 FOR j=1 TO c
730 IF p$(j)=i$ THEN LET found=j
740 NEXT j
750 IF found=0 THEN PRINT "ANIMALE
    NON TROVATO. RIPROVARE": GOTO 700
760 FOR j=1 TO r
770 IF n(j,found)>0 THEN PRINT
    c$(j);"□";n(j, found)
775 NEXT j
780 PRINT'"PREMERE UN TASTO PER
    TORNARE AL MENU"
785 PAUSE 0
790 RETURN
```

```
699 REM # # OPZIONE 3 # #
700 PRINT "IMMETTERE NOME
    DELL'ANIMALE"
705 PRINT "PER OTTENERE UN ELENCO DEI
    POSSESSORI"
710 INPUT P$
715 PRINT
720 FOR J=1 TO C
730 IF PET$(J)=P$ THEN FOUND=J
```

3. In questa matrice si può vedere a colpo d'occhio il numero di concessionari d'auto per ogni marca e per ciascuna nazione

```
740 NEXT J
750 IF FOUND=0 THEN PRINT "ANIMALE
    NON TROVATO. RIPROVARE": GOTO 700
760 FOR J=1 TO R
770 IF N(J,FOUND)>0 THEN PRINT
    CH$(J);"□";N(J,FOUND)
775 NEXT J
780 PRINT "▣PREMERE UN TASTO PER
    TORNARE AL MENU"
785 GET A$:IF A$="" THEN 785
790 RETURN
```

```
699 REM **OPZIONE 3**
700 PRINT""IMMETTERE NOME
    DELL'ANIMALE"
705 PRINT "PER OTTENERE UN ELENCO DEI
    POSSESSORI"
710 INPUT P$
715 PRINT"
720 FOR J=1 TO C
730 IF PET$(J)=P$ THEN FOUND=J
740 NEXT
750 IF FOUND=0 PRINT "ANIMALE NON
    TROVATO. RIPETERE":GOTO 700
760 FOR J=1 TO R
770 IF N(J,FOUND)>0 PRINT
    CH$(J);"□";N(J,FOUND)
775 NEXT
780 PRINT""PREMERE UN TASTO PER
    TORNARE AL MENU"
785 A=GET
790 RETURN
```

```
699 REM **OPZIONE 3**
700 PRINT@99,"IMMETTERE NOME
    DELL'ANIMALE"
705 PRINT "PER OTTENERE UN ELENCO DEI
    POSSESSORI□";
710 INPUT P$
715 PRINT:PRINT
720 FOR J=1 TO C
730 IF PET$(J)=P$ THEN FOUND=J
740 NEXT
750 IF FOUND=0 THEN PRINT
    "ANIMALE NON TROVATO.
    RIPROVARE":GOTO 700
760 FOR J=1 TO R
770 IF N(J,FOUND)>0 THEN PRINT
    CH$(J);"□";N(J,FOUND)
775 NEXT
780 PRINT:PRINT "PREMERE UN
    TASTO PER TORNARE AL MENU"
785 IF INKEY$="" THEN 785
790 RETURN
```

Questa routine è più complessa e merita particolare attenzione. La linea 710 memorizza l'immissione nella variabile P$ (i$ sullo Spectrum). Supponiamo che si digiti "GATTO", cosicché P$ = "GATTO" (o i$ = "GATTO"). Le linee da 720 a 740 scorrono l'intera lista di animali cercando il nome "GATTO". Se lo trovano, alla variabile TROV viene assegnato il numero della colonna corrispondente. Nel nostro caso si ha TROV = 2, poiché GATTO è nella colonna 2. Se, invece, il computer raggiunge la linea 750 con TROV pari a zero (valore assegnato nella linea 300), allora il nome digitato non è nella lista e si passa a una nuova richiesta.

Le linee da 760 a 775 passano in rassegna ogni elemento della colonna 2: se viene trovato un elemento maggiore di zero, la persona corrispondente alla riga in esame possiede uno o più gatti e il nominativo è visualizzato assieme al numero di gatti posseduti.

Si veda se si riesce a seguire, sul listato,

cosa accade a ogni passo.

L'opzione 4 serve per ottenere un elenco degli animali posseduti da una persona, immettendone il nominativo. Il procedimento è identico a quello precedente, con la differenza che adesso il computer analizza una particolare riga della matrice, anziché una sua colonna.

```
799 REM **OPZIONE 4**
800 PRINT"“IMMETTERE IL NOME"
805 DIM f$(5): INPUT LINE f$
810 PRINT f$; "□POSSIEDE I SEGUENTI
    ANIMALI:"
815 PRINT"
820 FOR j=1 TO r
830 IF c$(j)=f$ THEN LET found=j
840 NEXT j
850 IF found=0 THEN PRINT"“NOMINATIVO
    NON TROVATO. RIPROVA": GOTO 800
860 FOR j=1 TO c
870 IF n(found,j)>0 THEN PRINT
    n(found,j);"□";p$(j)
875 NEXT j
880 PRINT"“PREMERE QUALSIASI TASTO
    PER TORNARE AL MENU"
885 PAUSE 0
890 RETURN
```

```
799 REM # # OPZIONE 4 # #
800 PRINT "IMMETTERE IL NOME"
805 PRINT "ELENCO DEGLI ANIMALI
    APPARTENENTI A"
810 INPUT F$
815 PRINT
820 FOR J=1 TO R
830 IF CH$(J)=F$ THEN FOUND=J
840 NEXT J
850 IF FOUND=0 THEN PRINT
    "NOMINATIVO NON TROVATO. RIPROVA":
    GOTO 800
860 FOR J=1 TO C
870 IF N(FOUND,J)>0 THEN PRINT;N
    (FOUND,J);"□";PET$(J)
875 NEXT J
880 PRINT "PREMERE QUALSIASI TASTO
    PER TORNARE AL MENU"
885 GET A$:IF A$="" THEN 885
890 RETURN
```

```
799 REM **OPZIONE 4**
800 PRINT"“IMMETTERE IL NOME"
805 PRINT "ELENCO DEGLI ANIMALI
    APPARTENENTI A"
810 INPUT F$
815 PRINT"
820 FOR J=1 TO R
830 IF CH$(J)=F$ THEN FOUND=J
840 NEXT
850 IF FOUND=0 PRINT"“NOMINATIVO NON
    TROVATO. RIPROVA": GOTO 800
860 FOR J=1 TO C
870 IF N(FOUND,J)>0 PRINT;N(FOUND,J);
    "□";PET$(J)
875 NEXT
880 PRINT"“PREMERE QUALSIASI TASTO
    PER TORNARE AL MENU"
885 A=GET
890 RETURN
```

```
799 REM **OPZIONE 4**
800 PRINT@99,"IMMETTERE IL NOME"
805 PRINT "ELENCO DEGLI ANIMALI
    APPARTENENTI A"
810 INPUT F$
815 PRINT:PRINT
820 FOR J=1 TO R
830 IF CH$(J)=F$ THEN FOUND=J
840 ENXT
850 IF FOUND=0 THEN PRINT:PRINT
    "NOMINATIVO NON TROVATO. RIPROVA":
    GOTO 800
860 FOR J=1 TO C
870 IF N(FOUND,J)>0 THEN PRINTN
    (FOUND,J);"□";PET$(J)
875 NEXT
880 PRINT:PRINT "PREMERE QUALSIASI
    TASTO PER TORNARE AL MENU"
885 IF INKEY$="" THEN 885
890 RETURN
```

L'opzione 5 serve per visualizzare un elenco di tutti coloro che possiedono un numero di animali pari o maggiore a quello immesso. Ecco la relativa subroutine:

```
899 REM **OPZIONE 5**
900 PRINT'"IMMETTERE UN NUMERO PER
    ELENCARE TUTTI COLORO CHE
    POSSIEDONO ALMENO ALTRETTANTI
    ANIMALI"
910 INPUT a
915 PRINT"
920 FOR j=1 TO r
925 FOR k=1 TO c
930 LET p(j)=p(j)+n(j,k)
935 NEXT k
940 IF p(j)>=a THEN PRINT c$(j);"□";
    p(j): LET found=1
945 LET p(j)=0
950 NEXT j
955 IF found=0 THEN PRINT"
    "NESSUNO HA□";a;"□O PIÙ ANIMALI"
960 PRINT"“PREMERE QUALSIASI TASTO
    PER TORNARE AL MENU"
970 PAUSE 0
980 RETURN
```

```
899 REM # # OPZIONE 5 # #
900 PRINT "IMMETTERE UN NUMERO PER
    ELENCARE TUTTI COLORO CHE
    POSSIEDONO ALMENO ALTRETTANTI
    ANIMALI"
910 INPUT A
915 PRINT
920 FOR J=1 TO R
925 FOR K=1 TO C
930 PT(J)=PT(J)+N(J,K)
935 NEXT K
940 IF PT(J)>=A THEN PRINT CH$(J);"□";
    PT$:FOUND=1
945 PT(J)=0
950 NEXT J
955 IF FOUND=0 THEN PRINT
    "NESSUNO HA"; A; "O PIÙ ANIMALI"
960 PRINT "■PREMERE QUALSIASI TASTO
    PER TORNARE AL MENU"
970 GET A$:IF A$="" THEN 970
980 RETURN
```

```
899 REM **OPZIONE 5**
900 PRINT'"IMMETTERE UN NUMERO PER
    ELENCARE"'"COLORO CHE POSSIEDONO
    ALMENO ALTRETTANTI ANIMALI"
910 IMPUT A
915 PRINT"
920 FOR J=1 TO R
925 FOR K=1 TO C
930 PT(J)=PT(J)+N(J,K)
935 NEXT K
940 IF PT(J)>=A THEN PRINTCH$(J);"□";
    PT(J): FOUND=1
945 PT(J)=0
950 NEXT J
955 IF FOUND=0 PRINT"“NESSUNO HA□";
    A;"□O PIÙ ANIMALI"
960 PRINT"“PREMERE QUALSIASI TASTO
    PER TORNARE AL MENU"
970 A=GET
980 RETURN
```

```
899 REM **OPZIONE 5**
900 PRINT@64,"IMMETTERE UN NUMERO
    PER ELENCARE TUTTI COLORO CHE
    POSSIEDONO ALMENO ALTRETTANTI
    ANIMALI"
910 INPUT A
915 PRINT:PRINT
920 FOR J=1 TO R
925 FOR K=1 TO C
930 PT(J)=PT(J)+N(J,K)
935 NEXT K
940 IF PT(J)>=A THEN PRINTCH$(J);"□";
    PT(J):FOUND=1
945 PT(J)=0
950 NEXT J
955 IF FOUND=0 THEN PRINT:PRINT
    "NESSUNO HA"; A; "O PIÙ ANIMALI"
960 PRINT:PRINT "PREMERE QUALSIASI
    TASTO PER TORNARE AL MENU"
```

```
970 IF INKEY$="" THEN 970
980 RETURN
```

Quali problemi comportano i cicli nidificati?

L'impiego di matrici serve anche a impratichirsi con le istruzioni IF ... THEN e i cicli FOR ... NEXT. La parte più delicata, nell'uso delle matrici, è strutturare con precisione i cicli FOR ... NEXT nidificati: è facilissimo sbagliarsi e ingarbugliare le righe con le colonne. Si dia un'occhiata alle linee che servono alla lettura dei dati per la matrice, nella prima parte del programma (pagine 270-271). Ogni linea di DATA corrisponde a una riga della matrice, perciò il ciclo di lettura delle righe è quello esterno. Se le linee di DATA contenessero invece le colonne della matrice, allora quello esterno sarebbe il ciclo di lettura per le colonne. Ciò vale sia che la matrice venga indicata con N(R,C), o con N(C,R,): l'ordine degli indici C e R non ha importanza, purché rimanga lo stesso in tutto il programma. Le intestazioni di righe e colonne si preparano più accortamente con cicli a parte.

Prima di operare un confronto tra il numero immesso e il totale per ogni nominativo, occorre calcolare questo totale: le linee 920 e 925 predispongono i cicli per scorrere le righe e le colonne della matrice, mentre la linea 930 calcola il totale di animali per ogni riga. I totali vengono raccolti nella matrice PT() o, sullo Spectrum, p(). Se il totale per ogni riga è maggiore o uguale al numero immesso, il corrispondente nominativo viene visualizzato, con accanto il numero di animali posseduti.

Se viene trovato almeno un nominativo, il "flag" TROV è posto a 1. Ma se, giunti alla linea 955, TROV vale ancora zero, vuol dire che nessun bambino risponde al requisito immesso e tale evento viene segnalato con un messaggio.

L'ultima opzione, la 6, visualizza la matrice sullo schermo. Siccome è difficile far entrare i nomi degli animali in cima alla tabella, al loro posto compaiono dei numeri. Al contrario, i nomi dei possessori vengono visualizzati per intero.

```
999 REM **OPZIONE 6**
1000 PRINT '"NOMINATIVI", "ANIMALI"
1010 PRINT 'TAB 9;
1015 FOR j=1 TO c
1020 PRINT j;"□□";
1025 NEXT j: PRINT
1030 FOR j=1 TO r
1035 PRINT 'c$(j);TAB 9;
1040 FOR k=1 TO c
1050 PRINT n(j,k);"□□";
1060 NEXT k
1065 NEXT j
1070 PRINT ''"PREMERE QUALSIASI TASTO
   PER TORNARE AL MENU"
1080 PAUSE 0
1090 RETURN
```

```
999 REM # # OPZIONE 6 # #
1000 PRINT "NOMINATIVI",,"ANIMALI▪"
1010 PRINT TAB(6);
1015 FOR J=1 TO C
1020 PRINT J;SPC(2);
1025 NEXT J:PRINT
1030 FOR J=1 TO R
1035 PRINT CH$(J);TAB(6);
1040 FOR K=1 TO C
1050 PRINT N(J,K);SPC(2);
1060 NEXT K:PRINT:NEXT J
1070 PRINT"▪PREMERE QUALSIASI TASTO
   PER TORNARE AL MENU"
1080 GET A$:IF A$=""THEN 1080
1090 RETURN
```

Per adattare il programma allo schermo del Vic 20, sono necessarie due modifiche alla versione per il Commodore 64:

```
1020 PRINT J;
1050 PRINT N(J,K);
```

La routine inizia visualizzando le intestazioni principali "NOMINATIVI" e "ANIMALI", poi numera gli animali da 1 a 7. Se nella propria statistica si ha un numero di colonne diverso da 7, le spaziature alle linee 1020 e 1050 vanno cambiate. Le linee successive visualizzano la matrice una riga per volta, con a sinistra il nome del bambino e in ogni colonna il numero di animali.

```
999 REM**OPZIONE 6**
1000 PRINT'"NOMINATIVI","ANIMALI"
1010 PRINT'TAB(9);
1015 FOR J=1 TO C
1020 PRINT;J;SPC(4);
1025 NEXT
1030 FOR J=1 TO R
1035 PRINT' CH$(J);TAB(9);
1040 FOR K=1 TO C
1050 PRINT;N(J,K);SPC(4);
1060 NEXT K,J
1070 PRINT''"PREMERE QUALSIASI TASTO
   PER TORNARE AL MENU"
1080 A=GET
1090 RETURN
```

```
999 REM **OPZIONE 6**
1000 PRINT@32,"NOMINATIVI","ANIMALI"
1010 PRINT@72,;
1015 FOR J=1 TO C
1020 PRINTJ;
1025 NEXT:PRINT
1030 FOR J=1 TO R
1035 PRINT:PRINT CH$(J)TAB(8);
1040 FOR K=1 TO C
1050 PRINT N(J,K);
1060 NEXT K,J
1070 PRINT:PRINT:PRINT"PREMERE
   QUALSIASI TASTO PER TORNARE AL
   MENU"
1080 IF INKEY$="" THEN 1080
1090 RETURN
```

ADATTAMENTO DELLA MATRICE

Talvolta, si usa indicare una matrice impiegata in questo modo col termine *database*. Si possono creare database per ogni sorta di archiviazione. Un esempio potrebbe essere una statistica sul mondo animale, raccogliendo dati relativi a un gruppo di animali osservati e al loro ambiente. Se viene usata una matrice per vari periodi dell'anno o per diverse zone d'osservazione, si potranno ricavare statistiche sulla distribuzione delle specie.

Un'altra applicazione potrebbe riguardare un database che serva a elencare la quantità di grassi, carboidrati, proteine e calorie in diversi tipi di cibo. Si potrebbero così distinguere facilmente tutti i cibi che possiedono più di una certa quantità di proteine, o che hanno meno di un certo numero di calorie o, ancora, esaminare la composizione di un particolare alimento.

Un uso ancor più immediato delle matrici è la registrazione delle merci in un magazzino. In tal caso, le righe e le colonne possono riferirsi a scaffali e a posizioni reali degli oggetti, mentre il dato rappresenta la quantità in giacenza. Il programma, che dovrà prevedere opzioni per l'aggiornamento dei dati in base alle vendite e agli acquisti, potrà essere impiegato per visualizzare un avviso, se il livello delle scorte scende al di sotto di un minimo prefissato. Ma, a questo punto, ognuno avrà già in mente le applicazioni più adatte per le proprie esigenze.

COME IMMETTERE IL CODICE MACCHINA

- TROVARE UNA COLLOCAZIONE AL CODICE MACCHINA
- I MONITOR PER CODICE MACCHINA
- USO DEI MONITOR
- ESECUZIONE DEI PROGRAMMI

All'accensione, il computer entra direttamente in BASIC. Può essere utile, quindi, un programma per l'inserimento e l'esecuzione di programmi in codice macchina

Se non si possiedono appositi programmi per la stesura e l'esecuzione di programmi in codice macchina (quali il package CHAMP, fornito in dotazione a INPUT), la programmazione in codice macchina presenta uno strano paradosso. Infatti, nonostante il codice macchina sia quello usato dallo stesso microprocessore, occorre un programma in BASIC per immetterlo ed eseguirlo. Attraverso delle POKE, il codice macchina va depositato, un byte alla volta, nella memoria del computer. Per di più, sullo Spectrum, sullo ZX81 e sui Commodore non si può usare direttamente la notazione hex, ma solo quella decimale, obbligando a una ulteriore conversione di per sé inutile.

Negli Acorn, invece, è addirittura incorporato un 'assembler' per la conversione delle istruzioni in codice macchina. Quanto esposto in queste pagine, dunque, concerne particolarmente chi possiede un Dragon o un Tandy (per i quali non è fornita una versione di CHAMP adatta) o chi desideri includere sezioni di codice macchina nei propri programmi BASIC.

COME COLLOCARE IL CODICE MACCHINA

Una scelta importante, prima di immettere un programma in codice macchina, è dove collocarlo. Infatti, non lo si può mettere in un'area qualsiasi rischiando di interferire con locazioni riservate al funzionamento dello stesso computer.

Anche nell'adoperare l'area BASIC occorre andare cauti. Siccome una routine in codice macchina può essere richiamata da un programma in BASIC, questo potrebbe sovrapporsi al codice macchina durante l'esecuzione, compromettendo il funzionamento dell'intero programma.

Con programmi di piccole dimensioni, si possono usare aree quali il buffer per le cassette, purché il programma non preveda l'impiego del registratore.

Nello Spectrum, ad esempio, si può utilizzare l'area riservata alla grafica definita dall'utente (tra FF58 e FFFF nel modello 48K), se non è previsto l'uso di UDG. In genere, però, i programmi in codice macchina vengono collocati tra l'area UDG e il termine dell'area BASIC e intervenendo sul valore del puntatore al margine superiore della memoria (RAMTOP). L'area BASIC, non potendo oltrepassare RAMTOP, non rischia così di sovrapporsi al codice macchina.

RAMTOP è una variabile di sistema, il cui valore è contenuto nelle locazioni di memoria 5CB2 e 5CB3. Per accorciare l'area BASIC, basta depositare in queste due locazioni un valore più basso. Tuttavia il BASIC dello Spectrum possiede un comando CLEAR, col quale si ottiene in sostanza il medesimo effetto. Basta farlo seguire dall'indirizzo decimale al quale si desidera far scendere RAMTOP.

CLEAR azzera il *display file* come un CLS e tutte le variabili, riassegna la posizione di PLOT a Ø,Ø in basso a sinistra, ripristina il valore iniziale del puntatore alle DATA e pulisce lo stack, ponendolo subito sotto il nuovo valore di RAMTOP. Su uno Spectrum 16K si usa il comando:

```
CLEAR 31999
```

per abbassare RAMTOP a 31.999 (ossia 7CFF in hex), riservando così 6ØØ byte (tra RAMTOP e fino al termine dell'area UDG 32.6ØØ) ai programmi in codice macchina. Sul modello 48K si usa il comando:

```
CLEAR 63999
```

Anche questo serve per abbassare RAMTOP, ma stavolta a 63.999 (o F9999 in hex). I programmi in codice macchina possono quindi iniziare da 64.ØØØ (FAØØ in hex), lasciando più dello stretto necessario per la maggior parte delle esigenze. Probabilmente si occuperà uno spazio minore di quello riservato, ma è sempre meglio abbondare un po' nel caso occorresse in seguito correggere o ampliare il sottoprogramma.

Naturalmente, per programmi in codice macchina più lunghi, si può abbassare ulteriormente RAMTOP. In teoria, si potrebbe arrivare fino a CLEAR 23821, ma in pratica un limite così basso di RAMPTOP non lascerebbe spazio sufficiente neppure per una sola linea di BASIC.

Sullo ZX81, RAMTOP può essere abbassata intervenendo sui valori conservati nelle locazioni di memoria 16.388 e 16.389, riservando così un'area ai programmi in codice macchina, senza il rischio che essi vengano sovrapposti dal BASIC. Il problema con questo metodo è che non consente di trascrivere su nastro i programmi conservati in tale zona di memoria.

In alternativa, si possono depositare brevi programmi in codice macchina (lunghi meno di 32 byte), nel buffer della stampante, che va dalla locazione 16.444 alla 16.476, purché non sia previsto l'impiego della stampante.

Esistono anche altre piccole aree in cui depositare i propri programmi in codice macchina, ma anche in tal caso non li potremmo poi conservare. Sullo ZX81, quindi, non rimane altro che usare l'area dei programmi del BASIC, pur mantenendoli protetti da sovrapposizioni. Possono venir immessi sotto forma di frasi REM, di matrici o di stringhe di caratteri. In genere il primo sistema risulta il migliore.

Per immettere un programma mediante frasi REM, ne va prima calcolata la lunghezza in byte. Poi, nella prima linea del proprio programma in BASIC, va inserita una REM, seguita da un certo numero di caratteri, ad esempio punti o asterischi il cui numero deve essere almeno pari al numero dei byte del programma in codice macchina. Quale sia il carattere usato per riempire la linea REM è irrilevante, ma conviene adoperare una sequenza di caratteri identici, per meglio individuare la fine del programma in codice macchina, esaminando la memoria. A questo punto, nelle locazioni di memoria così preparate, va inserito il programma in codice macchina, un byte alla volta. Se la linea REM è la prima del programma BASIC, l'area interessata inizia dalla locazione 16.514. Per creare spazio protetto si può anche dimensionare una matrice. La linea BASIC:

RAMTOP
BASIC
BASIC

10 DIM A(1ØØ)

riserva 5ØØ locazioni di memoria, nell'area delle variabili. Per ogni elemento della matrice, lo ZX81 libera cinque locazioni atte a contenere numeri a virgola fluttuante. In questo caso, per verificare da dove iniziare la memorizzazione del codice macchina, occorre leggere il valore della variabile di sistema VARS, situata nelle locazioni 16.4ØØ e 16.4Ø1. Basta un comando diretto del tipo:

PRINT PEEK 164ØØ + PEEK 164Ø1 + 6

Si ottiene così l'indirizzo iniziale dell'area protetta creata, dal quale iniziare l'inserimento, byte per byte, del proprio programma. Nel comando, all'indirizzo viene sommato 6 per oltrepassare altrettante locazioni riservate alla matrice. Un'altra variante è creare spazio per una stringa con una linea di BASIC del tipo:

10 LET S$ = "..............."

con un numero di punti superiore a quello dei byte nel programma in codice macchina. Anche questo spazio protetto compare all'inizio dell'area delle variabili e se ne trova l'indirizzo iniziale sommando sei al valore che si ottiene con PEEK VARS: le sei locazioni, stavolta, contengono il nome della stringa.

Il problema con questi due ultimi metodi è che, premendo CLEAR o eseguendo un programma BASIC, l'area delle variabili viene ripulita, perdendo così il programma in codice macchina.

Sul Commodore 64, l'area di memoria da CØØØ a CFFF in hex (da 49.152 a 53.247 in decimale), è riservata al codice macchina e di norma quest'area è sufficiente per la maggior parte dei sottoprogrammi. Nel Vic 2Ø non esiste un'area riservata.

Programmi brevi, non più lunghi di 191 byte, si possono depositare, sia col C 64 che col Vic 2Ø, nel buffer del registratore, purché non si intenda adoperarlo. Questo buffer va dalla locazione Ø33C alla Ø3FB (hex) o da 828 a 1.Ø19 (decimale).

Se occorre più spazio, si può accorciare l'area BASIC, lasciandovi al di sopra un'area protetta, come fa il comando CLEAR su altri apparecchi. Sul Commodore 64 ciò si ottiene depositando, ad esempio il valore 32, nelle locazioni 52 e 56. L'area BASIC viene così abbassata a 8.192, liberando 32.768 byte per il codice macchina, non sovrapponibili dal BASIC, al quale restano circa 6.144 byte.

Fatto ciò, si digita CLR (le tre lettere C, L e R e non il tasto CLR col quale si comprometterebbe l'eventuale programma contenuto nell'area BASIC).

Le locazioni di memoria 52 e 56 contengono le parti "alte" degli indirizzi che stabiliscono il limite superiore dell'area BASIC. I byte "bassi", o meno significativi, degli indirizzi, contenuti in 51 e 55, valgono di solito Ø. Perciò, depositando il valore 32 nelle locazioni 52 e 56, si sposta il puntatore al limite superiore dell'area BASIC a 32×256 = 8.192. L'area BASIC può venire completamente annullata depositando in 52 e in 56 il valore 8, che abbassa il limite superiore a 2.Ø48, ossia al fondo della stessa area. Si potrebbe usare un valore Ø per le POKE, ma in tal caso il Commodore non è più in grado di far niente per l'impossibilità di immettere qualsiasi linea di BASIC.

Alternativamente, si può alzare il fondo dell'area BASIC e depositare il codice macchina nello spazio così riservato. Ciò si ottiene depositando opportuni valori nelle locazioni 43 e 44. Questi sono i byte relativi ai puntatori che segnano il fondo dell'area BASIC. Quest'ultimo potrebbe essere addirittura portato a 8.192, depositando in 43 il valore 1 e in 44 il valore 32, rendendo necessario fissare un nuovo indirizzo al BASIC (locazione 8.192) depositandovi uno zero e liberando così abbastanza spazio per routine in codice macchina, sia sopra che sotto al BASIC.

Dopo di ciò va digitato NEW, altrimenti si rischia di compromettere il funzionamento di eventuali programmi nell'area BASIC.

Sul Vic, RAMTOP può essere abbassato in quanto permette l'espansione di memoria in uso. Con PEEK 44 si legge dov'è il fondo dell'area BASIC; il valore trovato (aumentato di 1 o 2) lo si usa nelle POKE, nelle locazioni 52 e 56, per lasciare un po' di spazio al BASIC.

Per proteggere programmi in codice macchina, si può semplicemente usare il comando CLEAR, che abbassa RAMTOP. In tal modo si riserva parte del BASIC, subito sopra RAMTOP. CLEAR, inoltre, azzera tutte le variabili numeriche e alfanumeriche. Il formato del comando di solito è:

CLEAR 2ØØ,3ØØØØ

Si abbassa così RAMTOP da 32.767 decimale (&H7FFF in hex), a 3Ø.ØØØ, liberando 2.767 byte per programmi in codice macchina. Su questi computer CLEAR svolge anche un'altra funzione: l'assegnazione di spazio per le stringhe, la cui quantità è data dal primo numero dopo la CLEAR. All'accensione del computer, questo riserva 2ØØ byte (subito sotto RAMTOP) per le stringhe, uno spazio sufficiente per molte applicazioni, che può restare invariato anche quando si abbassa artificialmente RAMTOP. Il limite RAMTOP può scendere quasi fino al limite dell'area grafica, anche se un po' di spazio va pur sempre lasciato al BASIC per eseguire il comando. Su Dragon e Tandy anche lo stesso limite dell'area grafica può venir spostato mediante l'istruzione PCLEAR.

Con PCLEAR 1 si lascia una sola della pagine grafiche, cosicché la CLEAR può arrivare, più o meno, a &HC4Ø. Con PCLEAR 8, invece, si abilitano tutte le otto pagine grafiche, per cui la CLEAR piò arrivare appena a &H368Ø. Non usando il comando PCLEAR, il computer assegna automaticamente quattro pagine grafiche, così con CLEAR si può liberare fino a &H1E8Ø.

Questi limiti dipendono per lo più da ciò che si ha in memoria: basta la presenza di un qualsiasi programma in BASIC per provocare un messaggio di esaurimento di memoria. Perciò, conviene non avvicinarsi troppo ai limiti massimi, dato che a un certo punto servirà pure un programma in BASIC con cui richiamare quello in codice macchina. In genere, capita raramente di dover andare oltre 3Ø.ØØØ, con cui si liberano 2.767 byte.

Se si vuole iniziare il programma in codice macchina a un indirizzo "tondo", come ad esempio 3Ø.ØØØ, va digitato:

CLEAR 2ØØ,29999

Il secondo numero dopo la virgola specifica l'indirizzo più alto accessibile ai programmi in BASIC, per cui le locazioni libere per programmi in codice macchina iniziano da quella subito superiore.

MONITOR PER IL CODICE MACCHINA

I seguenti programmi permettono di immettere il codice macchina, di conservare poi i programmi su nastro e di rileggerli in memoria. Prima di immettere il codice macchina, va impartito un appropriato comando CLEAR, dopodiché, trascritto e avviato il programma "monitor", compare la richiesta dell'indirizzo iniziale. Con Spectrum, ZX81, Tandy e Dragon tale indirizzo deve corrispondere a quello specificato con la CLEAR, aumentato di 1.

Sui Commodore, il codice macchina deve iniziare da CØØØ, salvo che il programma superi i 4.Ø95 byte, nel qual caso occorre passare nell'area BASIC, previo deposito (con POKE) del valore 32 nelle locazioni 52 e 56. L'indirizzo iniziale della routine in codice macchina diviene 8.192 decimale. Per il Vic 2Ø, l'inizio dei programmi in codice macchina dipende dall'espan-

2017 91 FB
2016 88
2014 B1 FB
2012 A0 Ø1
2010 85 FD
200E B1 FB
200C A0 ØØ
200A 85 FE
2008 A8 ØØ
RAMTOP
1FF7 2A
1FF6 4F
1FF5 4B
1FF2 2E 4D 4B
1FEF 2C 2F 2C
4C 3A
4B
4C 4C
4B

sione di memoria in uso (si veda sopra). Con il Commodore non occorre aggiungere 1 all'indirizzo iniziale del codice macchina, perché la varizione dei valori contenuti nelle locazioni 52 e 56 tiene già conto di ciò. Adesso si trascriva il *monitor* adatto al proprio computer:

```
5 POKE 23658,8
10 PRINT INVERSE 1;AT5,6;"□
   MONITOR PER SPECTRUM□"
15 PRINT AT 8,4;"1:—□IMMISSIONE□
   CODICE MACCHINA"
20 PRINT AT 10,4;"2:—□VISIONE□
   MEMORIA"
30 PRINT AT 12,4;"3:—□SCRITTURA□
   BYTES□SU□NASTRO"
70 LET A$ = INKEY$: IF A$<"1" OR A$>
   "3" THEN GOTO 70
80 CLS : GOSUB 100+200*(VAL A$—1)
90 RUN
100 INPUT"INDIRIZZO DI PARTENZA?□";SA
110 INPUT LINE D$
120 IF D$="" THEN GOTO 110
125 IF D$(1)="#" THEN RETURN
127 IF LEN D$=1 THEN GOTO 110
130 LET S$=D$(1 TO 2)
140 FOR N=1 TO 2
150 IF S$(N)>"9" THEN LET S$(N)=CHR
   $ (CODE S$(N)—7)
155 IFS$(N)>"?" OR S$(N)<
   "0" THEN PRINT FLASH 1;"CARATTERE□
   NON□VALIDO": GOTO 110
160 NEXT N
170 POKE SA, (CODE S$(1)—48)*16+
   CODE S$(2)—48
180 PRINT SA,D$( TO 2)
190 LET D$=D$(3 TO )
200 LET SA=SA+1
210 GOTO 120
300 INPUT "INDIRIZZO□DI□PARTENZA?□";
   SA
310 INPUT "STAMPANTE□(S/N)?□"; LINE P
   $
320 LET ST=2: IFP$="S" THEN LET ST=3
330 PRINT #ST;SA;
340 FOR M=0 TO 7
350 LET H$="##"
360 LET H$(1)=CHR$ (INT (PEEK (SA
   +M)/16)+48)
370 LET H$(2)=CHR$ (48+PEEK (SA+M)
   —16*(CODE H$(1)—48))
380 FOR N=1 TO 2
390 IF H$(N)>"9" THEN LET H$(N)=CHR
   $ (CODE H$(N)+7)
400 NEXT N
410 PRINT #ST;TAB 7+3*M;H$;
420 NEXT M
440 LET SA=SA+8
445 PRINT #ST: POKE 23692,0
450 LET A$=INKEY$: IF A$="" THEN
   GOTO 450
460 IF A$=CHR$ 13 THEN RETURN
470 GOTO 330
500 INPUT "INDIRIZZO DI PARTENZA?□";SA
510 INPUT "NUMERO DI BYTES?□";N
520 INPUT "NOME DEL FILE?□"; LINE N$
530 IF LEN N$<1 OR LEN N$>10 THEN
   GOTO 520
540 SAVE N$CODE SA,N
550 RETURN
```

Prima di avviare questo programma è necessario immettere una REM alla linea 1, seguita dal numero eastto di caratteri per contenere il programma in codice macchina (un carattere per byte di codice macchina):

```
10 PRINT AT 4,9;"MONITOR ZX81"
15 PRINT AT 8,4;"1:—IMMISSIONE CODICE
   MACCHINA"
20 PRINT AT 10,4;"2:—VISIONE MEMORIA"
50 LET A$=INKEY$
60 IF A$<"1" OR A$>"2" THEN GOTO 50
70 CLS
80 GOSUB 100+200*(VAL A$—1)
85 CLS
90 RUN
100 PRINT AT 21,0;"INDIRIZZO DI
   PARTENZA?"
105 INPUT SA
110 INPUT D$
120 IF D$="" THEN GOTO 110
125 IF D$(1)="$" THEN RETURN
130 LET S$=D$(1 TO 2)
135 SCROLL
140 FOR N=1 TO 2
150 IF S$(N)<"0" OR S$(N)>"F" THEN
   PRINT "INVALID CHARACTER"
155 IF S$(N)<"0" OR S$(N)>"F" THEN
   GOTO 110
160 NEXT N
170 POKE SA, (CODE S$(1)—28)*16+
   CODE S$(2)—28
175 SCROLL
180 PRINT SA,D$(TO 2)
190 LET D$=D$(3 TO)
195 IF LEN D$=1 THEN GOTO 110
200 LET SA=SA+1
210 GOTO 120
300 PRINT AT 21,0;"INDIRIZZO DI
   PARTENZA?"
310 INPUT SA
320 SCROLL
330 PRINT SA;
340 FOR M=0 TO 7
350 LET H$="□□"
360 LET H$(1)=CHR$ (INT (PEEK (SA+M)/
   16)+28)
370 LET H$(2)=CHR$ (28+PEEK (SA+M)
   —16*(CODE H$(1)—28))
400 PRINT TAB 7+3*M;H$;
410 NEXT M
440 LET SA=SA+8
450 LET A$=INKEY$
460 IF A$="" THEN GOTO 450
470 IF A$=CHR$ 118 THEN RETURN
480 GOTO 320
```

```
5 POKE 53280,0:POKE 53281,0:HH$="0123
   456789ABCDEF"
10 PRINT "◘■■■■■■■◣"TAB
   (8)"1:—□IMMISSIONE□CODICE□
   MACCHINA"
20 PRINT TAB(8)"■■2:—□□□VISIONE
   □MEMORIA"
30 PRINT TAB(8)"■■3:—□SCRITTURA□
   BYTES□SU□NASTRO"
40 GET A$:IF A$<"1" OR A$>"3" THEN
   40
50 ON VAL(A$) GOSUB 100,200,300
60 GOTO 10
100 INPUT "◘INDIRIZZO DI PARTENZA□";
   SA
110 INPUT "◙■■□□□□□▌▌▌▌▌
   ▌▌▌";D$:IF D$=""THEN 110
120 IF D$="#" THEN RETURN
125 W=0:FOR Z=1 TO 16:IF LEFT$(D$,1)
   =MID$(HH$,Z,1) THEN W=W+1
126 IF RIGHT$(D$,1)=MID$(HH$,Z,
   1) THEN W=W+1
127 NEXT:IF W<2 THEN 110
130 A=ASC(D$)—48:B=ASC(RIGHT$(D$,
   1))—48
140 C=B+7*(B>9)—(LEN(D$)=2)*(16*(A
   +7*(A>9)))
150 POKE SA,C:PRINT "◙■■■■■□
   □□□□□□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□□
   □□□▼"SA,D$
160 SA = SA + 1:GOTO 110
200 INPUT "◘INDIRIZZO□DI□PARTENZA";
   SA
210 INPUT "STAMPANTE□(S/N)□";P$:
   PRINT "◘"
220 IF P$="S" THEN OPEN 4,4:CMD4
230 PRINT SA;:FOR M = 0 TO 7
240 A = (PEEK(SA)/16):PRINT MID$ (HH$,A
   +1,1);MID$(HH$,PEEK(SA)—INT(A)*16
   +1,1)"□";
250 SA = SA + 1:NEXT
260 IF P$="S" THEN PRINT#4,"";:CLOSE4
270 GET A$:IF A$="" THEN 270
280 IF A$=CHR$(13) THEN RETURN
290 PRINT:GOTO 220
300 CLR:INPUT "◘INDIRIZZO□DI□
   PARTENZA";A:A=A—3:AA=INT(A/256):
   A2=A—AA*256
310 INPUT "INDIRIZZO□FINALE□□";B:B=B
   +1:BB=INT(B/256):B2=B—BB*256
315 INPUT "NOME□DEL□FILE□□□";N$
320 PRINT"◘P◧44,"AA":P◧43,"A2
330 PRINT "■■P◧46,"BB":P◧45,"B2:
   PRINT"■■SCRITTURA"CHR$(34)N$
   CHR$(34)
```

```
340 PRINT"■■■■■■■■■P□
   44,"PEEK(44)":P□43,"PEEK(43)
350 PRINT"■■P□46,"PEEK(46)":P□45,
   "PEEK(45):PRINT"■■ESECUZIONE■"
```

(C=)

Per far funzionare la versione per il Commodore 64 sul Vic 20, occorre cambiare queste linee:

```
5 POKE 36879,8:HH$="0123456789ABCDE
  F"
10 PRINT "■■■■■■■■■■1:—
  □IMMISSIONE CODICE MACCHINA"
20 PRINT "■■2:—□□□VISIONE□
  MEMORIA"
30 PRINT "■■3:—□SCRITTURA BYTES□
  SU NASTRO"
150 POKE SA,C:PRINT "■■■■■■□
  □□□□□□□□□□□□□□□□□□□
  □□□■"SA,D$
230 PRINT SA;:FOR M=0 TO 4
```

(Dragon/Tandy)

```
10 CLS
20 PRINT@196,"1:—□IMMISSIONE□CODICE
  □MACCHINA"
30 PRINT@260,"2:—□VISIONE□MEMORIA"
40 PRINT@324,"3:—□SCRITTURA□BYTES
  □SU□NASTRO"
50 A$=INKEY$:IF A$<"1" OR A$>
  "3" THEN GOTO 50
60 CLS
70 ON VAL(A$) GOSUB 200,400,600
80 GOTO 10
200 INPUT "INDIRIZZO□DI□PARTENZA□";
  SA
210 LINE INPUT D$
220 IF D$="" THEN GOTO 210
230 IF LEFT$(D$,1)="#" THEN RETURN
240 S$=LEFT$(D$,2)
250 POKE SA,VAL("&H"+S$)
260 PRINTSA,LEFT$(D$,2)
270 D$=MID$(D$,3)
280 SA=SA+1:GOTO 220
400 INPUT "INDIRIZZO□DI□PARTENZA□";
  SA
410 PRINT"STAMPANTE□(S/N)□?"
420 A$=INKEY$:IF A$="" THEN 420
430 P=0:IF A$="S" THEN P=-2
440 PRINT #P,SA;
450 FOR M=0 TO 7
460 PRINT #P,"□";RIGHT$("□"+
  HEX$(PEEK(SA+M)),2);
470 NEXT:PRINT #P
480 SA=SA+8
490 A$=INKEY$:IF A$="" THEN 490
500 IF A$=CHR$(13) THEN RETURN
510 GOTO 440
600 INPUT "INDIRIZZO□DI□PARTENZA□";
  SA
610 INPUT "NUMERO□DI□BYTES□";N
620 LINE INPUT"NOME□DEL□FILE□";N$
630 CSAVEM N$,SA,SA+N,SA
640 RETURN
```

COME FUNZIONA IL MONITOR

Quando si esegue un RUN del programma, compare un menu che chiede se si vuole immettere codice macchina, esaminare la memoria o (tranne che sullo ZX81) scrivere una porzione di memoria su nastro. Desiderando immetterre codice macchina (il che presuppone l'aver preparato un programma in tale codice), si prema [1]. Dopo aver fornito l'indirizzo di partenza, e premuto [ENTER] o [RETURN], si inizia l'immissione del codice macchina sotto forma di cifre hex da immettere due caratteri per volta.

Si possono immettere quante coppie si vogliono prima di premere [ENTER] o [RETURN], ma è meglio digitare una linea per volta e controllare attentamente le cifre prima di immetterle: è molto più facile correggerle quando sono sullo schermo che non quando sono già in memoria. Sul Commodore, dopo ogni hex di due cifre va premuto [RETURN].

Si noti che i programmi per lo Spectrum, lo ZX81 e i Commodore ritraducono i numeri hex in decimale prima di depositarli in memoria, dal momento che la POKE del BASIC, su questi apparecchi, non accetta valori hex (si vedano le pagine 240-247). Tutte queste conversioni, apparentemente inutili, consentono invece al programmatore di lavorare nel formato hex, che è molto più comodo col codice macchina.

I programmi in codice macchina vanno conclusi con un segno # (o con $ sullo ZX81), onde tornare al menu. Selezionando l'opzione 2 (visione della memoria), viene ancora chiesto un indirizzo di partenza, ma stavolta relativo all'area che si vuole leggere. Per esaminare il codice macchina appena immesso, questo indirizzo deve essere identico a quello dato nell'opzione 1.

Poi, eccetto che sullo ZX81, viene chiesto se si vuole una copia su stampante del programma in codice macchina. Rispondendo con [N], il programma visualizza l'indirizzo d'inizio, i contenuti di quella locazione e le successive sette locazioni su una linea dello schermo.

Adesso, premendo un qualsiasi tasto, salvo [RETURN] o [ENTER], vengono visualizzate le otto locazioni successive, nello stesso formato precedente. In questa maniera si può visualizzare, linea per linea, l'intero programma.

Se si individua un errore, il sistema per correggerlo è il seguente: si prema [ENTER] o [RETURN], per riportare il programma al menu. Si scelga l'opzione 1 e, quando il computer chiede l'indirizzo di partenza, si immetta quello del byte sbagliato. Se un'intera serie è sbagliata, si dia l'indirizzo del primo byte della serie.

Qundi si immettano i byte corretti, col solito procedimento.

Se di errori ce n'è soltanto uno, lo si può correggere anche interrompendo il programma *monitor* e depositando il valore appropriato nella locazione interessata, mediante una POKE. Si rammenti che sullo Spectrum e sul Commodore il numero va espresso in decimale.

Nell'opzione 1, terminate le correzioni, si prema [ENTER] o [RETURN], poi si esamini di nuovo la memoria per assicurasi che stavolta vada bene.

CONSERVARE LE ROUTINE

Desiderando conservare il programma *monitor*, occorre un'operazione di SAVE distinta da quella per il codice macchina, adottando le normali procedure (vedere a pagina 23).

Infatti, l'opzione "Scrittura byte su nastro" si limita a conservare le routine in codice macchina immesse con questo programma.

Al contrario, sullo ZX81 le routine di codice macchina vengono conservate insieme al monitor nel modo usuale.

Sugli altri apparecchi, premendo [3], il programma chiede ancora una volta un indirizzo di partenza, che deve corrispondere a quello della routine in codice macchina appena immessa, benché si possa conservare qualsiasi parte di memoria specificandone l'indirizzo di partenza.

Le versioni di questo programma per lo Spectrum, il Dragon e il Tandy chiedono quindi la lunghezza in byte del codice macchina. Va poi assegnato un nome alla routine, in modo che il computer sappia identificarla quando, in seguito, la si vuole riutilizzare.

Si rammenti di azionare i tasti del registratore, secondo la consueta procedura di SAVE. Una volta completata la scrittura, ricompare il menu.

La versione del Commodore richiede un indirizzo finale, ottenibile sommando a quello di partenza il numero di byte del programma, oppure preoccupandosi di annotarselo durante la revisione della memoria (opzione 2).

Assegnato un nome alla routine, il programma visualizza una linea di programma, sullo schermo: si posizioni qui il cursore con [HOME] e si prema [RETURN] tre volte; si passi quindi alla procedura SAVE.

Sul Commodore, non è possibile riutilizzare il programma monitor, finché non vengono ripristinati i puntatori modificati dalla procedura di SAVE. Ciò si ottiene riportando il cursore all'inizio della linea di programma a metà schermo e poi premendo tre volte RETURN. Così facendo, si depositano i valori originali nei puntatori e si riesegue automaticamente il RUN.

LETTURA DEL CODICE MACCHINA

Per la routine in codice macchina, con lo Spectrum, il Dragon e il Tandy si usa la normale procedura LOAD (si veda a pagina 22), ma se si sta adoperando il programma monitor, occorre premere BREAK per uscire dal programma.

Se nel frattempo si è invece spento l'apparecchio o lo si è utilizzato per altri programmi, si rammenti di ripetere il comando CLEAR, in modo da proteggere il codice macchina.

Con i Commodore non si può usare un comando LOAD generico per ricaricare da nastro il codice macchina: ciò non garantisce che il codice torni nello stesso luogo da cui è stato copiato quando lo si è trasferito su nastro. Il comando LOAD deve avere questo formato:

LOAD "*Routinename*",1,1

"Nomeroutine" è il nome della routine in codice macchina che si vuole caricare e va sempre racchiuso tra doppi apici. Il primo 1 è il numero del dispositivo che corrisponde al registratore. Il secondo 1 comunica al Commodore di rimettere il programma nella stessa zona di memoria da cui è venuto.

Se si usano dischetti, il suffisso da indicare nel comando LOAD è, 8,1.

Inoltre, se si sono cambitati i puntatori del BASIC quando si è scritta e conservata la routine di codice macchina, occorre ora ripristinarli con gli stessi valori. Omettendo questa operazione, la routine non sarà protetta da eventuali sovrapposizioni da parte del BASIC.

Con lo ZX81, si può ricaricare da nastro il proprio programma in codice macchina insieme al monitor nel modo consueto.

SCROLL ALL'INDIETRO

Per fornire un esempio pratico, il seguente programma è in codice macchina e serve per ottenere uno scorrimento (scroll) dello schermo dall'alto verso il basso, anziché dal basso in alto. Si immetta la routine usando il programma monitor, ricordando, sullo Spectrum, lo ZX81, il Dragon e il Tandy, di non digitare spazi tra i vari numeri. Sui Commodore i numeri si immettono uno alla volta.

```
21 9F 58 11 BF 58 06 08
25 15 E5 D5 C5 01 A0 00
ED B8 06 02 C5 D5 11 00
F9 19 D1 01 20 00 ED B8
E5 21 00 F9 19 EB E1 01
E0 00 ED B8 C1 10 E5 C1
D1 E1 10 D4 21 00 3F 06
08 C5 24 E5 AF 77 54 5D
13 01 1F 00 ED B0 E1 C1
10 EF 21 9F 5A 11 BF 5A
01 A0 02 ED B8 21 00 58
11 01 58 3A 8D 5C 77 01
1F 00 ED B0 C9 #
```

```
01 18 03 2A 0C 40 09 54
5D 01 F7 02 2A 0C 40 09
ED B8 2A 0C 40 23 36 00
54 5D 13 01 1F 00 ED B0
C9
```

```
A9 13 20 D2 FF A9 11 20
D2 FF A9 9D 20 D2 FF A9
94 20 D2 FF A9 A0 85 DA
A9 0D 20 D2 FF 60 #
```

```
8E 05 E0 A6 84 A7 89 FE
00 30 88 E0 A6 84 A7 88
20 8C 04 00 2C F3 30 89
02 01 8C 06 00 25 E4 39
#
```

ESECUZIONE DEI PROGRAMMI

Una volta immesso il programma in codice macchina e dopo essersi assicurati (esamindolo in memoria) che non ci siano errori, lo si può eseguire.

Ma, dato che non si tratta di un programma BASIC, il comando RUN in questo caso non serve a niente. Occore invece un comando specifico.

Nello Spectrum e nello ZX81 si usa l'istruzione USR del BASIC, seguita dall'indirizzo del programma in codice macchina. USR ritorna al BASIC con il valore del registro BC, al completamento della routine in codice macchina. Tale valore non è utile in sé, ma indica che il programma è stato eseguito. Purtroppo, USR non è un comando, ma una funzione e la struttura del BASIC Sinclair richiede invece che una linea di programma inizi con un comando. Perciò, sullo Spectrum, USR va usato in modo simile a questo:

RANDOMIZE USR 32000

e sullo ZX81:

RAND USR 16514

(Si noti che il tasto RAND sullo Spectrum genera sullo schermo RANDOMIZE; sullo ZX81 il tasto RAND genera sullo schermo RAND.) In sé, questo comando non ha senso, ma serve ad eseguire la routine in codice macchina. A questo fine, la USR può essere preceduta da un qualsiasi comando BASIC: per esempio, PRINT USR 32000 visualizza sullo schermo il valore di BC.

RANDOMIZE USR 32000 (o RAND USR 16514) è quello più usato perché evita effetti collaterali, ma non va usato se il programma in codice macchina prevede l'impiego di numeri casuali, altrimenti si combina un pasticcio. In questo caso, sullo Spectrum si usi:

LET L=USR 32000

e sullo ZX81 si usi:

LET L=USR 16514

In tal modo alla variabile L viene assegnato il valore del registro BC non appena completata l'esecuzione della routine in codice macchina: non una gran cosa in sé, ma semplice da usare, dato che chi ha un Sinclair noterà che LET, L, e USR sono tutti sullo stesso tasto.

Il modo più semplice di eseguire un programma in codice macchina su Commodore 64 e Vic 20 è digitare SYS seguito dall'indirizzo d'inizio della routine in codice macchina. Se si vogliono però scambiare variabili tra il codice macchina e il BASIC si può usare la funzione BASIC USR.

La sintassi usata è:

A=USR (B) o PRINT USR (B) in modo diretto.

USR manda il computer alla routine in codice macchina che inizia all'indirizzo contenuto nelle locazioni 785 e 786 (nel consueto formato byte "alto"/byte "basso"). Occorre quindi depositare in tali locazioni l'indirizzo di partenza della routine che si vuole eseguire.

Il valore di B viene passato al programma in codice macchina attraverso il primo accumulatore a virgola fluttuante, che va dalla locazione 97 alla 102 (97 è l'esponente, da 98 a 101 è la mantissa e 102 riporta il segno). La USR ritorna anch'essa al BASIC col contenuto dell'accumulatore a virgola fluttuante, al termine della routine in codice macchina. Questo sistema permette di trasferire valori tra il codice macchina e il BASIC.

Su Dragon e Tandy ci sono due istruzioni per eseguire programmi in codice macchina. C'è il comando EXEC, a cui segue l'indirizzo iniziale della routine in codice macchina da eseguire. Per esempio:

```
EXEC24ØØØ
```

esegue la routine che inizia alla locazione 24.ØØØ decimale.

Il comando USR, invece, permette di scambiare variabili tra il programma in BASIC e la routine in codice macchina.

Innanzitutto occorre definire l'inizio della routine che interessa con l'istruzione DEFUSR. Questa va fatta seguire da un numero compreso tra Ø e 9, cosicché si possano definire fino a dieci routine USR, cioè si possano usare fino a dieci subroutine in codice macchina in ogni programma in BASIC. Se DEFUSR non è seguito da un numero, per il computer corrisponde a DEFUSRØ. La sintassi è:

```
1Ø DEFUSR1=24ØØØ
```

Si definisce così la routine USR numero 1 come avente inizio alla locazione 24.ØØØ decimale.

Il comando che provoca effettivamente l'esecuzione del codice macchina è USR. Un piccolo inconveniete della ROM del Dragon fa sì che una routine in codice macchina vada richiamata da USR seguito da |e dal numero precedentemente definito dalla routine. Sul Tandy non serve lo Ø.

La sintassi consueta è:

P=USRØ1 (Q) o PRINT USRØ1 (Q).

Il comando USR copia il valore della variabile Q nell'accumulatore a virgola fluttuante, dal quale può venir ripreso dalla routine in codice macchina.

Al termine di questa, i contenuti dell'accumulatore tornano al programma BASIC (in questo esempio, nella variabile Q). Allo stesso modo, si possono scambiare variabili stringa tra il BASIC e il codice macchina.

RELAZIONI... SENSATE!

I computer possono eseguire operazioni di logica per molti milioni di volte al secondo. Ma la logica è anche una parte fondamentale di ogni programma...

Nella programmazione BASIC si usano tre tipi di espressioni. Gran parte del lavoro riguarda espressioni aritmetiche o contenenti stringhe, ma è il terzo tipo, le espressioni *logiche*, che spesso svolge funzioni decisionali in un programma.

Il loro semplice scopo è quello di valutare se qualcosa è 'vero' o 'falso'. Parole e simboli del programma, usati a questo fine, si chiamano *operatori* (qualcuno li definisce anche *connettori* o connettivi).

Nelle espressioni logiche si usano due tipi di operatori: *operatori relazionali* (che comprendono simboli matematici come <,> e =) e *operatori logici*, che in tutte le forme di BASIC fanno parte della lista di parole chiave come AND, OR, NOT.

MAGGIORE, UGUALE, MINORE

Gli operatori relazionali sono usati nei più semplici programmi di BASIC e tramite loro i confronti possibili sono:

A >B ... A è maggiore di B
A <B ... A è minore di B
A > = B ... A è maggiore o uguale a B
A < = B ... A è minore o uguale a B
A = B ...A è uguale a B
A <> B ... A è diverso da B

Li abbiamo sicuramente già visti all'opera in molti programmi di INPUT o altrove e, anche se si possono usare in aritmetica pura, il loro valore in prove condizionali è evidente in connessione con IF ... THEN. Un altro esempio è:

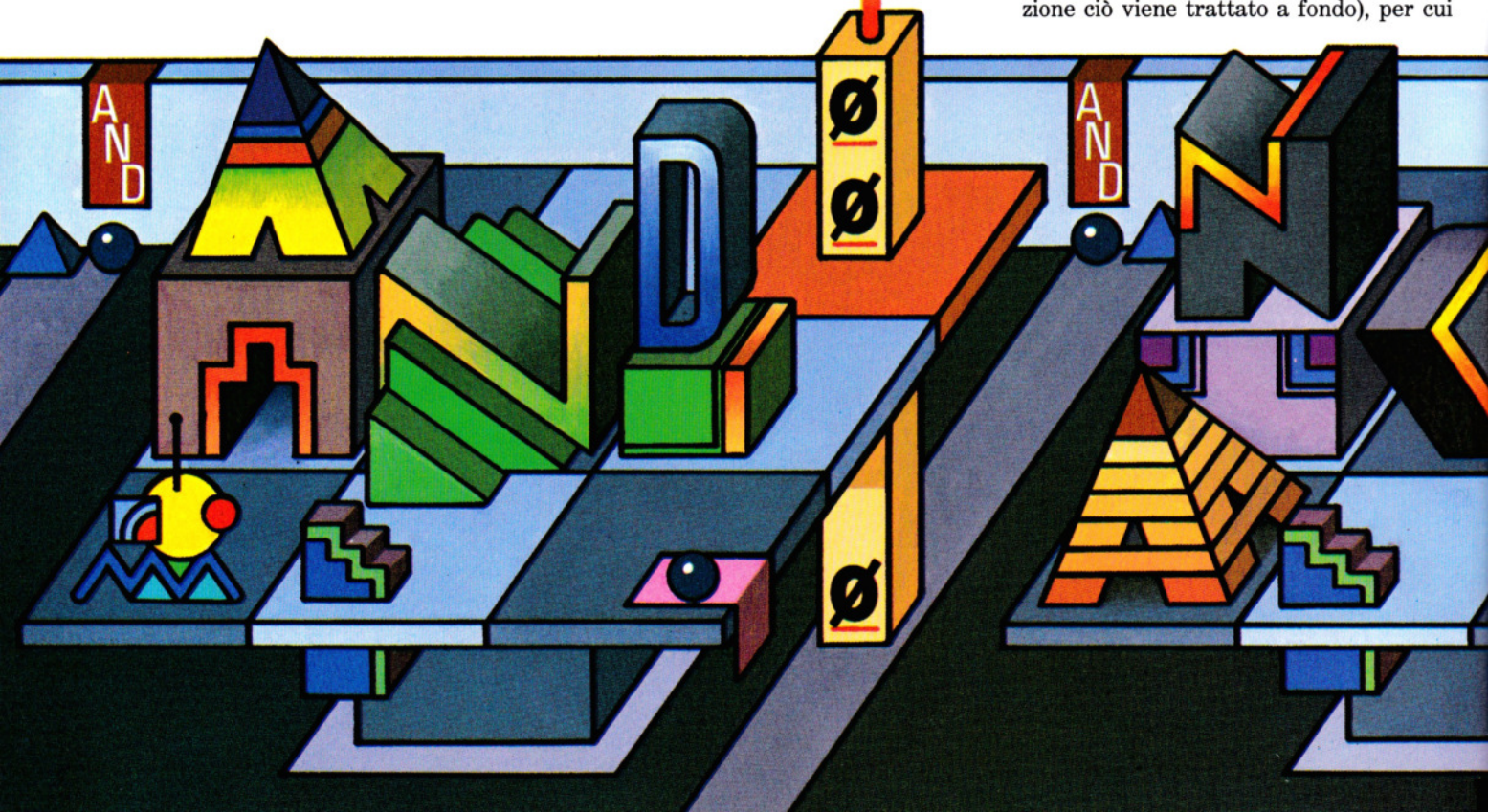
```
IF A>B THEN PRINT "A È MAGGIORE DI B"
```

Il valore di A *deve* superare quello di B affinché il resto della linea sia eseguito. Quando, invece, il valore di A rimane al di sotto di quello di B, il programma passa semplicemente alla linea di programma successiva. Si capisce come ciò renda possibile qualche diramazione condizionale: se una serie di valori è vera, il programma fa una cosa; se è vera un'altra serie di valori, allora fa qualcos'altro.

Gli operatori relazionali non sono limitati solo a valori numerici, ma si usano anche nel confronto tra stringhe, purché attuato con attenzione onde evitare risultati incongruenti. Il primo punto da ricordare è che il confronto si ferma sempre al numero di caratteri della più breve delle due stringhe a confronto: per cui, se una stringa contiene undici caratteri e l'altra sette, solo i primi sette della stringa più lunga sono confrontati con i corrispondenti nella stringa più breve.

Il confronto avviene, in realtà, un carattere alla volta, da sinistra a destra ed è questo fatto che può generare strani risultati. In un cofronto *numerico* la proposizione 5>10 è ovviamente falsa, ma in un confronto di *stringhe* può esserci una proposizione in cui due variabili sono confrotate nella forma A$>B$. Se A$="5" e B$="10", il computer prima confronta il 'carattere' a sinistra da ogni lato, 5 e 1 e poi si ferma, ritenendo che non ci sia altro da confrontare.

Erroneamente, si ottiene una condizione 'vera', poiché 5 è maggiore di 1, ma il computer ignora l'importanza delle cifre restanti nella stringa più lunghe. Attenzione a questi tipo di errori!

In stringhe di caratteri alfabetici, le singole lettere hanno il seguente ordine crescente: "A"<"B"<"C"<"D" e così via. Ma occorre fare ancora attenzione, perché il computer non riconosce realmente il significato dei caratteri: si limita a confrontare i *codici* dei caratteri (in un'altra lezione ciò viene trattato a fondo), per cui

- OPERATORI RELAZIONALI
- DECIDERE SE QUALCOSA È MAGGIORE, MINORE O UGUALE A QUALCOS'ALTRO

- OPERATORI LOGICI
- USO DI **AND, OR** E **NOT**
- LE TAVOLE DELLA VERITÀ
- USO DI **EOR** SUGLI ACORN
- OPERAZIONI SUI BIT

hanno un significato anche gli spazi e gli altri simboli diversi dalle lettere.

Questo fatto può causare errori madornali in programmi che debbano ordinare alfabeticamente una serie di stringhe.

Se due stringhe contengono una stessa seguenza di caratteri, allora viene considerata numericamente più grande la stringa più lunga. Però, una stringa più breve viene considerata più grande in un'espressione del tipo "ABD">"ABCD" perché il confronto cessa alla prima differenza, ossia di fronte ai valori dei codici di "D" e "C". In pratica, il criterio alfabetico è molto simile a quello adottato nei dizionari o negli indici, nei quali "anno" viene prima di "annotare", ma dopo "anacronismo".

VERO O FALSO

Il risultato di un qualsiasi confronto è un numero intero: il risultato è Ø se il confronto è falso e (a seconda dell'apparecchio) –1 o 1 se è vero. Si immetta il seguente comando in *modo diretto* (ossia senza numero di linea), per controllare sul proprio computer:

```
PRINT 6>5, 5>7
```

L'espressione a sinistra è vera, quella a destra falsa, per cui sullo schermo si visualizzerà un –1 (o 1) e uno Ø.

Gli interi che esprimono il risultato si possono direttamente usare nei programmi, per eseguire calcoli. Attenzione però alle divisioni: viene segnalato un errore se si tenta di dividere qualcosa per Ø.

OPERATORI LOGICI

Le parole chiave AND, OR e NOT, chiamate *operatori logici*, estendono la portata decisionale di IF ... THEN (vedere a pagina 33). Per esempio, se (IF) la condizione 1 è vera e (AND) è vera la condizione 2, allora (THEN) esegui una certa azione. In modo analogo, vengono prese decisioni anche più complesse. Questi operatori, detti anche *Booleani* (dal matematico Boole), si possono usare sia in modo diretto che nei programmi, per confrontare numeri o stringhe o per ottenere un "valore di verità" di alcune (spesso complesse) verifiche condizionali.

In pratica, consentono di semplificare espressioni condizionali, che sarebbero altrimenti un vero intrico di istruzioni IF ... THEN.

L'USO DI AND

L'operatore AND si può considerare con lo stesso significato della congiunzione 'e'. In una espressione come:

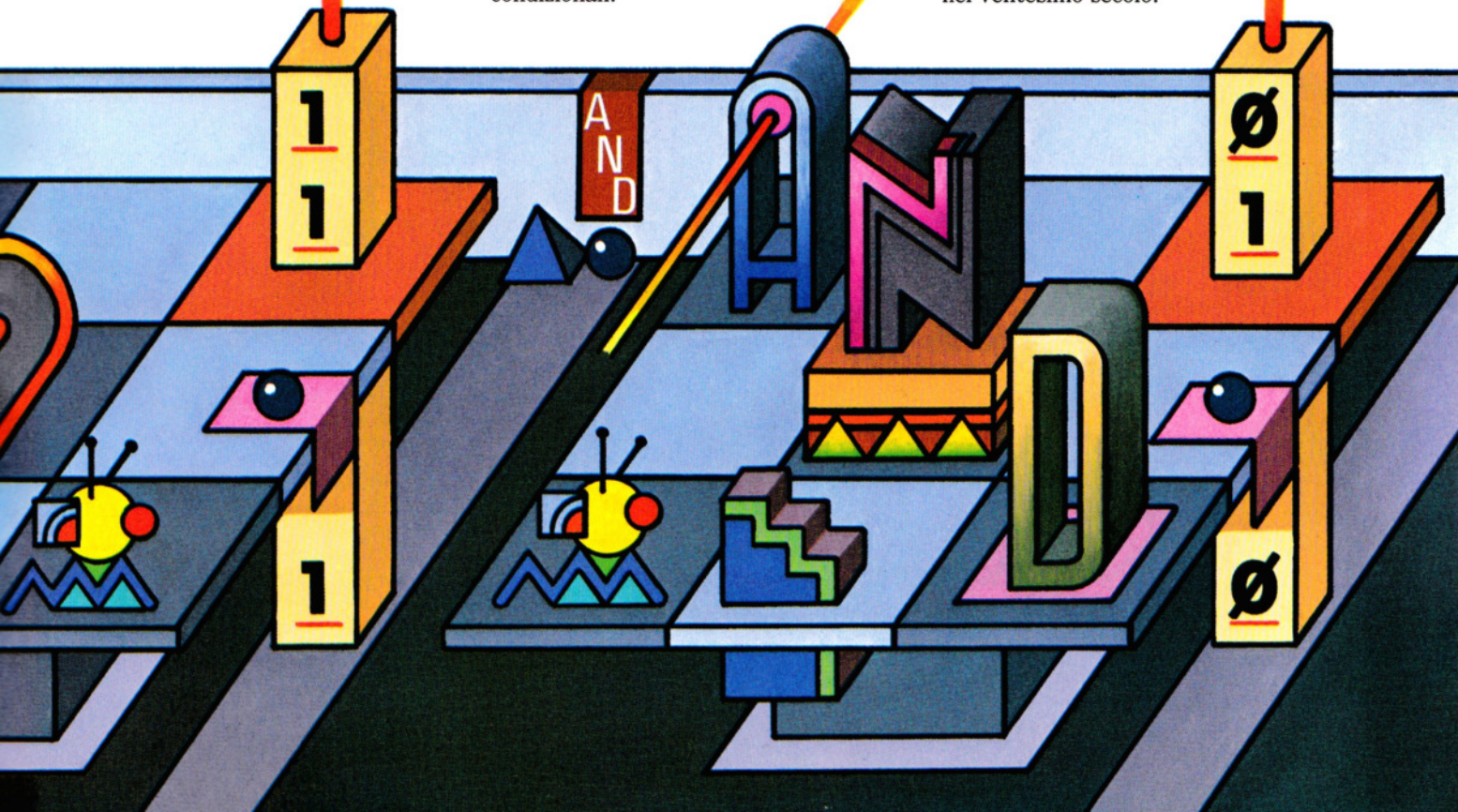
```
IF V>Ø AND V<1ØØ PRINT "VALIDO"
```

il messaggio compare soltanto se V è maggiore di Ø e minore di 1ØØ.

In un programma si può usare una linea come questa:

```
990 OKAY=1 AND MESE>5 AND MESE<
    10 AND ANNO>1900 AND ANNO<2001
```

```
990 OKAY=–1 AND MESE>5 AND MESE
    <10 AND ANNO>1900 AND ANNO<
    2001
```

Questa linea può esser utile per controllare se una particolare data cade d'estate nel ventesimo secolo.

Qui AND viene usata in quattro verifiche, *ognuna* delle quali deve essere vera affinché rimanga vero anche OKAY. In questa "prova di verità", la variabile OKAY è dapprima posta a 'vero' (a −1, ma sul Sinclair a 1). Poi si verifica che il MESE sia compreso tra 6 e 9 e l'ANNO tra il 1901 e il 2000. In altre parole, se tutte le condizioni si avverano, l'espressione ha un valore vero. In sostanza, la linea di programma si traduce in:

```
990 OKAY = -1 AND-1 AND -1 AND
-1 AND-1
```

(sul Sinclair, 1 invece di −1).

Basta scrivere PRINT OKAY per controllare che il risultato sia −1, ossia "vero".

L'USO DI OR

OR corrisponde alla 'o' disgiuntiva del linguaggio comune, anche se non del tutto. Osserviamo come viene usata in una linea di programma dove si confrontano i valori di una variabile:

```
IF V=8 OR V=10 PRINT "OKAY"
```

Il messaggio compare se V è 8 *oppure* 10.

OR è di grande utilità nel controllae la validità di un immissione da tastiera: se, per esempio, viene chiesta l'immissione di un'età, può accadere che si digiti −10 o 999. Si può aggirare il problema così:

```
10 INPUT A
20 IF A<1 OR A>120 THEN PRINT "NON
   È POSSIBILE": GOTO 10
```

L'USO DI NOT

Il terzo operatore logico comunemente usato è NOT, leggermente diverso dagli altri, in quanto si applica soltanto all'espressione numerica o logica che segue: mentre AND e OR confrontano espressioni sui due lati, NOT opera solo su un singolo valore. NOT si può usare in una linea di programma come:

```
IF NOT (A>10) THEN 999
```

Il che equivale a dire "se il valore A *non* è maggiore di 10, si passi alla linea 999".

La funzione logica di NOT è la conversione di una condizione vera in una falsa, e viceversa. Si può usare NOT come commutatore o per ribaltare il valore vero/falso risultante da un'altra verifica.

LE TAVOLE DELLA VERITA'

Occupandosi di operatori logici, ci si trova spesso a contatto con le "tavole della verità". Queste rappresentano, visivamente e sinteticamente, cosa accade quando i valori vero (−1 o 1) e falso (0) sono messi in relazione da AND o OR. Non è necessaria una tavola per NOT, in quanto i valori sono semplicemente invertiti.

Le tavole di verità possono prendere forme diverse. Una tipica per AND è:

A	B	C	
−1	−1	−1	(linea 1)
−1	0	0	(linea 2)
0	−1	0	(linea 3)
0	0	0	(linea 4)

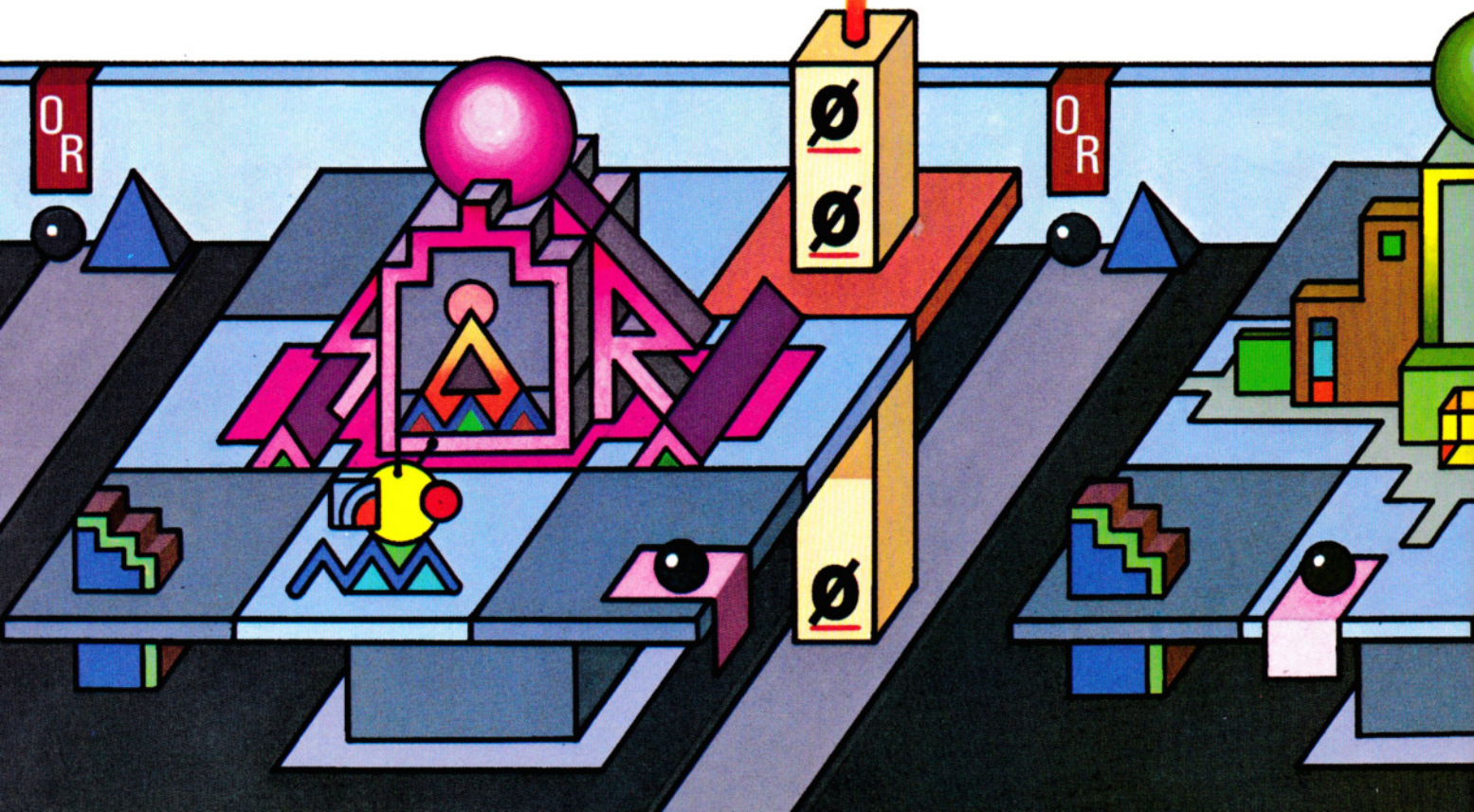
La tavola si legge interpretando la linea 1 come: "Se A è vero e B è vero, allora C è vero". La linea 2 significa "Se A è vero e B è falso, allora C è falso". La linea 3: "Se A è falso e B è vero, C è falso". La linea 4: "Se A e B sono entrambi falsi, allora C è falso". La tavola di verità per OR è:

A	B	C	
−1	−1	−1	(linea 1)
−1	0	−1	(linea 2)
0	−1	−1	(linea 3)
0	0	0	(linea 4)

La prima linea si legge: "Se A è vero e B è vero, allora C è vero". Le linee 2 e 3 possono essere entrambe interpretate come: "Se uno soltanto tra A o B è vero, allora C è vero". La linea 4: "Se A e B sono entrambi falsi, allora C è falso".

IN PRATICA

Ecco infine un programma che usa AND, OR e NOT. Il programma mostra come sia possibile suddividere in categorie i candidati a un concorso:

```
10 INPUT "ANNI"; ETA
20 INPUT "PUNTEGGIO"; GRAD
30 OLTR18 = (ETA> = 18)
40 QUAL=(GRAD> =70)
50 IF NOT OLTR18 AND NOT QUAL THEN
   PRINT "NON IDONEO"
60 IF (NOT OLTR18 AND QUAL) OR (OLTR18
   AND NOT QUAL) THEN PRINT "SECONDA
   CATEGORIA"
70 IF OLTR18 AND QUAL THEN PRINT
   "PRIMA CATEGORIA"
```

Supponiamo di immettere 2Ø per l'età e 6Ø per il punteggio: allora (ETÀ>18) è vero, ma (GRAD>=7Ø) è falso. Per cui il candidato è OLTR18, ma non QUALificato. Il computer incontra questa serie di condizioni alla linea 6Ø e visualizza "SECONDA CATEGORIA". Il programma potrebbe essere ampliato, per verificare un più ampio numero di condizioni, per tener conto di altri requisiti necessari.

(Si noti che su alcuni Micro BBC, i punti e virgola alle linee 1Ø e 2Ø vanno sostituiti con virgole.)

L'USO DI EOR

I computer Acorn possiedono un quarto operatore, chiamato EOR (sigla che stà per OR Esclusivo: si dà cioè un caso o un altro, ma non i due assieme). Può essere utile in espressioni come questa:

```
IF bus<=2 EOR aut<=6 THEN PRINT
   "PUOI SALIRE SUL TRAGHETTO"
```

Con EOR evitiamo che il traghetto si sovraccarichi sia di pullman che di auto.

Ecco una tavola di verità per EOR:

A	B	C	
−1	−1	Ø	(linea 1)
−1	Ø	−1	(linea 2)
Ø	−1	−1	(linea 3)
Ø	Ø	Ø	(linea 4)

Come gli altri operatori, EOR può anche essere usato con valori numerici, in particolare per commutare alternativamente una condizione tra vero e falso. Supponiamo di avere una linea come:

```
S=S EOR 1
```

in un ciclo e che, inizialmente, si sia posto S uguale a 1. Al primo giro S diventa uguale a Ø, al secondo a 1, al terzo a Ø e così via. Un simile "interruttore" logico è utile in più occasioni. Per esempio, COLOR S scambierà COLOR Ø con 1; PRINT CHR$ 224 + S scambierà il carattere 224 con il 225, il che potrebbe servire in una sequenza di animazione.

OPERAZIONI NUMERICHE

L'uso degli operatori logici non si limita alle istruzioni IF ... THEN viste più sopra, ma può comprendere anche alcuni tipi di operazioni numeriche. Probabilmente abbiamo già incontro simili operazioni, senza comprenderle del tutto. In effetti, non sempre funzionano nel modo che è naturale aspettarsi. Si immetta questo comando in modo diretto:

```
PRINT 375 AND 47
```

Ci si aspetterebbe di ottenere come risultato 422, o semmai 37547, invece si ottiene 39.

Per spiegarne il motivo occorre approfondire un poco il funzionamento del computer e in particolare come vengano trattate le due espressioni ai lati di AND nell'esempio.

Per prima cosa le espressioni (375 e 47) sono convertite in interi a due byte. Eccone la forma binaria:

Binario a due byte di 375:
ØØØØØØØ1Ø111Ø111
Binario a due byte di 47:
ØØØØØØØØØØ1Ø1111

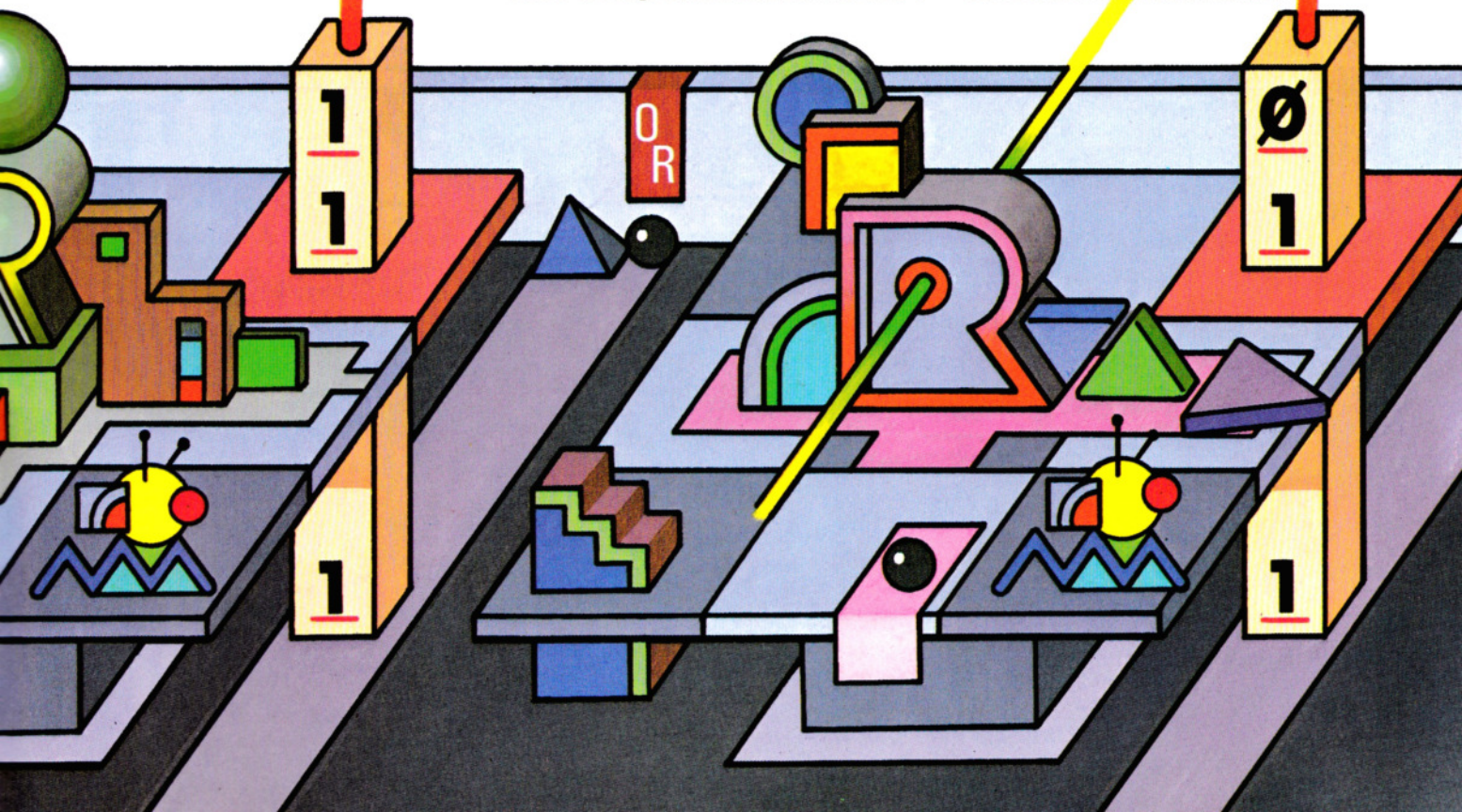
La AND confronta ad uno ad uno i 16 bit, producendo un 1 *solo* se si trova un 1 nella medesima posizione di ciascun numero binario.

Partendo da destra, solo i bit zero, uno, due e cinque soddisfano la condizione, producendo così il numero binario ØØØØØØØØØØ1ØØ111. Convertendo questo in decimale, si ottiene il valore 39: ecco dunque il risultato di 375 AND 47.

Ora si digiti questo in modo diretto:

```
PRINT 375 OR 47
```

Come si ottiene il risultato 378? L'operatore OR ha la proprietà di far porre a 1 il bit del risultato (binario) se *uno dei due* bit corrispondenti nei due operandi vale 1. Torniamo alla forma minaria a due byte dei numeri: i bit 8,6,5,4,3,2,1 e Ø hanno tutti un 1 in almeno uno dei due valori. Il numero binario risultante è dunque ØØØØØØØ1Ø1111111, il quale in decimale vale 383.

Il risultato di un'operazione OR può essere identico anche in espressioni diverse. Per esempio, anche PRINT 375 OR 75 dà come risultato 383. Per controllare, si convertano i numeri in binario.

Il valore −1 (memorizzato, per inciso, come FFFFFFFF hex, una sequenza di bit tutti posti a 1) resta inalterato quando OR lo pone in relazione con qualsiasi numero valido. Si provi questo:

PRINT −1 OR 375

e il risultato è −1.

Esaminiamo adesso un'applicazione numerica di NOT:

PRINT NOT 10.75, NOT −11

I risultati sono −11 e 10. L'operatore NOT, in pratica, aggiunge 1 al numero su cui opera, cambiandone il segno. Si noti che il numero a virgola fluttuante è arrotondato all'interno minore e che il valore reale si può stimare con X = −(X+1). In effetti, il valore è convertito in un intero di quattro byte, nel quale vengono invertiti tutti i bit.

Gli operatori logici sui Sinclair non operano confronti di numeri bit per bit perciò si usano di rado in operazioni numeriche.

OPERAZIONI SUI BIT

Sui computer Commodore, le operazioni logiche sui singoli bit hanno una funzione in più, poiché servono anche per il controllo di particolari attività del computer. Sui Commodore esistono alcune locazioni di memoria riservate, il cui valore controlla svariate funzioni: per selezionarne una, occorre una particolare sequenza di bit, tale da influire su alcuni dei bit, lasciando gli altri invariati. Per controllare i contenuti di una locazione di memoria si usano i comandi POKE e PEEK (vedere a pagina 246). Un comando tipico può essere:

POKE 1, PEEK (1) AND 251

che, sul Commodore 64, "spegne" il bit 3 della locazione di memoria 1. PEEK (1) dà 55 decimale ed ecco come opera la relazione AND:

Valore 55: 00110111
Valore 251: 11111011

La AND dà: 00110011 (=51). Confrontando la sequenza di bit del valore 55 con quello di 51 (nuovo valore PEEK della locazione 1), si comprende perché il bit 2 (il terzo da destra) viene cambiato in 0, ossia "spento".

Con OR si può ripristinare la situazione precedente:

POKE 1, PEEK (1) OR 4

L'unico 1 che compare nella forma binaria di 4 (00000100) dà 00110111, ossia il valore di PEEK originario, cioè 55.

Può sembrare una fatica sproporzionata, quando in molti casi si potrebbe direttamente depositare un valore con una POKE, senza ricorrere agli operatori logici. In questo caso, però, c'è il rischio di modificare il valore acceso/spento degli altri bit. Con OR basta indicare quale bit accendere (ossia porre a 1) e nient'altro viene alterato nella locazione di memoria.

AND serve invece a "spegnere" uno o più bit, semplicemente sottraendo il valore cumulativo di tutti i bit da spegnere da 255. Ad esempio, per spegnere i bit 0 e 2, che corrispondono a 5 decimale, si usa AND 250.

Per ripristinarne il valore originario si usa OR 5. Questo tipo di procedura viene talvolta indicato con l'espressione *mascheratura di bit*. La differenza principale consiste nel fatto che la POKE di un valore può influire sui restanti valori di una locazione, mentre la "mascheratura" interviene soltanto su quelli desiderati. Gli altri operatori logici non vengono usati a questi fini.

C'è un limite alla dimensione dei numeri da usare con AND e OR?

Nelle operazioni AND e OR, il Commodore 64, il Vic 20, il Dragon e il Tandy non possono usare numeri che superino il campo tra −32768 e +32767. Altrimenti, i Commodore segnalano un "errore di quantità", mentre i Dragon e i Tandy un "errore di funzione". Gli Acorn accettano numeri fino a quattro byte, da $-65536^2/2$ a $+65536^2/2-1$, quindi ogni operazione logica concepibile dovrebbe restare nel campo di validità. Sullo Spectrum non si possono operare AND e OR su valori numerici.

INDICE CUMULATIVO DEI FASCICOLI

A

Anagrammi, programma per 203
AND 35-36, 285-288
Animazione 26-32
ANGL, *Commodore 64* 88
Applicazioni
archivio per hobby 46-53, 75-79
bilancio familiare 136-143
grafici a barre 257-263
scrivere lettere 124-128
Archivio, programma per 46-53, 75-79
Assegnazione, istruzioni di 66-67, 92
Assembler, definizione 67
Assembly, linguaggio 66-67
ATTR, *Spectrum* 68-69
Avventure, giochi di, progetto 264-268

B

Basi numeriche 110-116
BASIC 65
BASIC, programmazione
cicli FOR...NEXT 16-21
grafica più sofisticata 184-192
i segnali del programmatore 60-64
immissione di dati 129-135
matrici 152-155
numeri casuali 2-7
operatori logici 284-288
PEEK e POKE 240-247
prendere decisioni 33-37
programmazione strutturata 173-178, 216-219
SIN e COS 250-256
stringhe 201-207
uso di PLOT, DRAW, LINE e PAINT 84-91
uso di READ e DATA 104-109
variabili 92-96
videate 117-123
BEEP, *Spectrum* 230-231
Binari, numeri 38, 41, 44, 45, 113-166
numeri negativi 179-183
Bit,
definizione 113
mascheratura di 288
Bubble sort, programma 216-219
Bussola, disegno di una 251-253
Byte, definizione 114

C

Campi 46, 75
Calcolatore, programma di conversione 112-113
Caratteri nella grafica, *Dragon, Tandy* 191-192
Carro armato, creazione e controllo 11-15
Carta per stampanti 228
Casa, disegno di una
Acorn 107-108
Commodore 64 108-109
Cassette, registratori a 25
Castello, disegno di un
Dragon, Tandy 108
Cerchio, disegno di un 255-256
CHR$, *Dragon, Tandy* 26-27
CIRCLE 86-91
CLS, spiegazione 27
CODE, *Spectrum* 8
Codice Macchina
architettura della macchina 236-239
esadecimali 156-160
immissione del 276-283
linguaggi di basso livello 65-67
mappe di memoria 208-215
nei giochi (grafica) 38-45
numeri binari 113-116
numeri negativi 179-183
numeri nonari 111-112
programmi monitor 276-283
ROM e RAM 208-215
un drago in 88-83
vantaggi del 66
velocizzare i giochi 8-15
COLOUR 87-90
Compilatori 66
Complemento a due 179-183
Contatempo, un semplice 176-177
Cursore, definizione codici di controllo,
Commodore 64, Vic 20 7, 123
COS, funzione 250-256
CPU 236-239

D

Dadi, lancio di 64
DATA 104-109
codice macchina 67
istruzione BASIC 8-14, 40-45
nella grafica 107-109
Database 275
Decimali 110
conversione da binario 38, 42
frazioni in binario 114
DEFPROC, *Acorn* 64
Diagrammi di flusso 173-178
DIM, *Dragon, Tandy* 41
Dimensionamento delle matrici 152-153
Disegno sullo schermo 132-133
DRAW 85-91

E

Effetti sonori 230-235
Elicottero, creazione di un 81
Ellissi, disegno di 256
ENDPROC, *Acorn* 64
EOR, *Acorn* 287-288
Errore, cause di 36
Esadecimali 38, 42, 45, 156-160
ESCAPE, *Acorn* 4

F

File, scrittura e lettura di 77
Flow chart 173-178
FOR...NEXT, cicli 16-21
Formattamento dello schermo 117-123

G

Giochi
alieni e missili 144-151
animazione 26-32
avventure 264-268
bombardamento 161-167
campo di mine 97-103
controllo del movimento 54-55, 57-59
caratteri in movimento 54-59
contapunti e contatempo 69-73, 97-103
effetti sonori 230-235
esplosione, grafica per 161-167
"fruit machine" 36
indovinelli 3-5
labirinti 68-74, 193-200, 230-235
sottoprogrammi 8-15
stazione spaziale 144-151
GET, *Commodore 64* 55, 132-134
GET$, *Acorn* 55-57, 58, 103, 132-134
GET, *Commodore 64* 135
Golf, disegno di un campo da
Acorn, Spectrum 184-191
GOSUB 62-64
GOTO 18-21, 60-62
Gradi e radianti, conversione 250-251
Grafica
bassa risoluzione 26-32
caratteri grafici 38-45
carro armato con UDG 10-15
creazione di UDG 8-15
dipingere coi numeri 19
disegnare al computer 107-109
drago sputafuoco 80-83
esplosioni, grafica per 161-167
grafica più sofisticata 184-192
rana con UDG 10-15
ricami e modelli 21
SIN e COS 250-256
tramonto al computer 20
uso di PLOT, DRAW, LINE, CIRCLE E PAINT
Acorn 88-90
Commodore 64 87-88
Dragon 90-91
Spectrum 85-86
Grafici, programma *Acorn* 64
Griglie per UDG 8-11

H

HIRES, *Commodore 64* 87

I

IF...THEN 3, 33-37
IF...THEN...ELSE 37
IF...THEN...GOTO 36, 54
INK, *Spectrum* 86
INKEY, *Acorn* 28-29, 103, 134-135
INKEY$ 54-55, 132-135
INPUT,
istruzione 3-5, 117-122, 129-135
INSTR, funzione 206

J

Joystick 220-224

L

Labirinti,
programmi per 68-75, 193-200
LEFT$, funzione 202-207
LEN, funzione 202-207
Lettere al computer 124-128
Linguaggi per computer 65
Assembly 66-67
BASIC 65
vedere: Codice Macchina
LINE, *Dragon, Tandy* 88-91
LOAD, comando 22-25

M

Margherita, stampanti a 227
Matrice di punti, stampanti a 226-277
Matrici bidimensionali 269-275
Memoria 208-215
Menu, uso di 46-47
Minuscole, per il *Dragon e Tandy* 142
Missili, lancio di 55-58
MID$, *Acorn* 202-207
MOVE, *Acorn* 71, 88-90
Movimento
MULTI, *Commodore 64* 87

N

NEW 10-15, 23
Numeri
binari negativi 180-183
casuali 2-7
dipingere coi 18
nonari 111
NOT, operatore logico 286-288

O

Opcode 67
Operatori logici 35, 284-288
Orologio interno 69-73
OR, operatore logico 35-36, 286-288

P

PAPER, *Spectrum* 86
Parametri 64
Parentesi, uso delle 35
Password, programma per 133
PAUSE
Commodore 64 88
Spectrum 101, 108
Pause nei programmi 17
PEEK 59, 101, 204-247
Periferiche,
registratori a cassette 22-25
joystick 220-224
stampanti 225-229
Pixel 84
PLAY, *Dragon, Tandy* 73, 234-235
PLOT 88-89
PMODE, *Dragon, Tandy* 12, 90
POINT, *Acorn* 71
POKE 101, 108-109, 240-247
Posizionamento del testo 117-123
Pressione dei tasti 54-55
PRINT 26-32, 117-123
PROCedure, *Acorn* 64
Programmazione strutturata 173-178, 216-219
PSET, *Dragon, Tandy* 13, 90-91
Punteggiatura nelle PRINT 119-123
Punteggio 97, 100-101

R

RAM 25, 44, 46, 208-215
Rana, creazione di una 10-15
RANDOMIZE 2
READ 40-44, 104-109
REC, *Commodore 64* 87
Record (elementi di file) 75-77
Registratori a cassette 22-25
REPEAT...UNTIL, *Acorn* 36
RESTORE 106-107
RETURN, istruzione 62
RIGHT$ 202-207
Risoluzione grafica 84
RND, funzione 2-7
ROM, grafica 26, 32, 107-109
Rubrica, programma per 105
RVS, *Commodore 64* 31

S

Satelliti, creazione di
Dragon 26-27
SAVE 22-25
Scenario innevato,
Commodore 64 186-188
SCREEN, *Dragon, Tandy* 40
Scroll all'indietro del video 282-283
Simboli aritmetici 6
Simon's BASIC, *Commodore 64* 87-88
SIN, funzione 250-256
SOUND,
Acorn, Dragon, Tandy 233-235
Spazi, uso degli, *Commodore* 122
Sprite, Commodore 64 14, 15-168-172
Stack, spiegazione 237-239
Stampante, scelta della 225-229
STEP 17-21
STOP, *Spectrum, ZX81* 4, 64
Stringhe
funzioni per 201-207
nulle 96
variabili alfanumeriche 4-5, 95-96
STRING$ 98-205
Subroutine 62-63

T

TAB 117-122
Tabelle di moltiplicazione 5-7
Teletext, grafica, *BBC* 28
Temporizzazione 97, 101-103

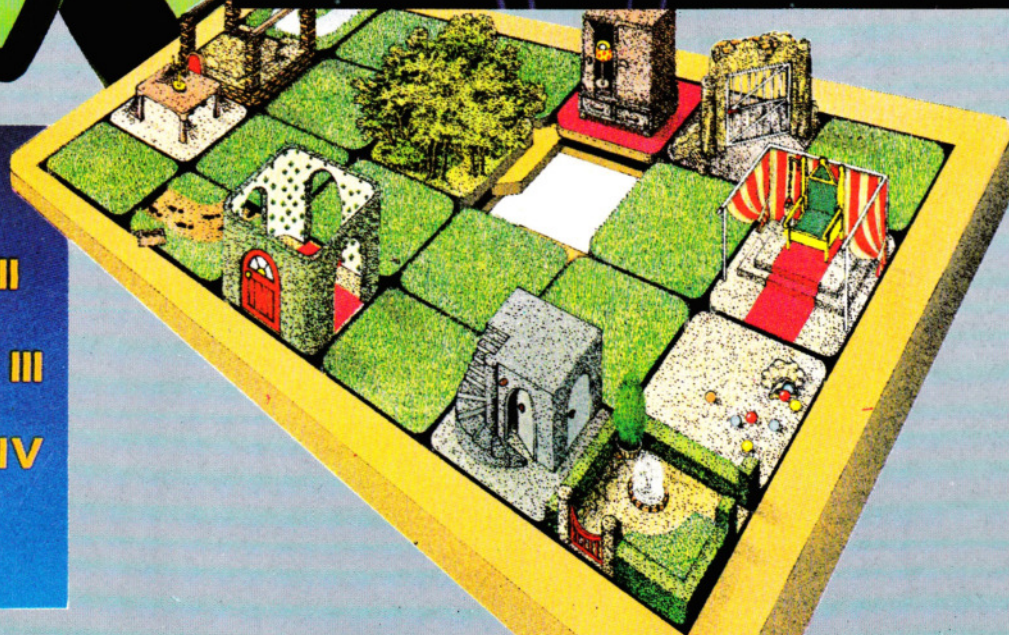
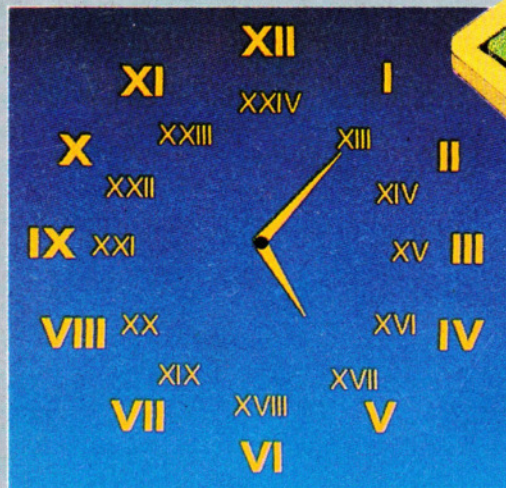
U

Uccello in volo, sprite,
Commodore 64 168-172
UDG
colorati, *Dragon, Tandy* 248-249
creazione di UDG 38-45
DATA per UDG 45
definizione 8-15, 40,44
griglie per UDG 8-11

V

VAL, *Commodore 64* 101
Variabili 3-5, 92-96, 104-108
VDU, *Acorn* 28-29, 70, 99
Verifica dei programmi registrati 24-25
VERIFY, comando 24
VIC, chip grafico,
Commodore 64 172

NEL PROSSIMO NUMERO

- ☐ *La prima di una serie di puntate dedicate allo sviluppo di un gioco d'avventura completo: LA MAPPA DI UN'AVVENTURA.*
- ☐ *Impariamo tutto sui CODICI ASCII: cosa sono, come usarli efficacemente nei programmi.*
- ☐ *Essere abili nell'uso della tastiera è un requisito importante anche per chi ha a che fare con la programmazione: ecco un programma di DATTILOGRAFIA AL COMPUTER.*
- ☐ *Impariamo le basi del LINGUAGGIO ASSEMBLY e come convertirlo in codice macchina.*
- ☐ *Le funzioni trigonometriche SIN e COS permettono di ottenere magnifici effetti grafici.*

CHIEDETE INPUT AL VOSTRO EDICOLANTE